VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

VYHLEDÁVÁNÍ GRAFFITI TAGŮ PODLE PODOBNOSTI

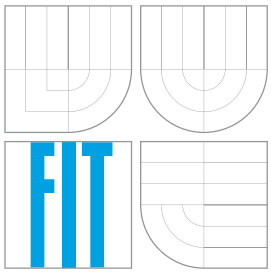
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

VOJTĚCH SEMERÁK

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

VYHLEDÁVÁNÍ GRAFFITI TAGŮ PODLE PODOBNOSTI

SIMILARITY BASED GRAFFITI SEARCH

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — VOJTĚCH SEMERÁK
AUTHOR

VEDOUCÍ PRÁCE — Doc. Ing. MARTIN DRAHANSKÝ, Ph.D.
SUPERVISOR

BRNO 2013

Abstrakt

Graffiti tag si lze představit jako podpis jeho autora. Tato práce zkoumá využití metod rozpoznávání podpisů v rámci prevence a potírání graffiti vandalismu. Taktéž jsou analyzovány rozdíly mezi podpisy a graffiti tagy.

Abstract

Graffiti tags are similar to the common human signatures. This report analyses, which signature recognition methods could be useful in the field of preventing graffiti vandalism. It also discusses differences between signatures and graffiti tags.

Klíčová slova

rozpoznávání podpisů, graffiti, tag, počítačové vidění

Keywords

signature recognition, graffiti, tag, computer vision

Citace

Vojtěch Semerák: Vyhledávání graffiti tagů podle podobnosti, diplomová práce, Brno, FIT VUT v Brně, 2013

Vyhledávání graffiti tagů podle podobnosti

Prohlášení

Prohlašuji, že jsem tuto práci vypracoval samostatně pod vedením pana docenta Drahanského.

.....
Vojtěch Semerák
21. května 2013

Poděkování

Rád bych tímto poděkoval panu docentu Drahanskému za cenné připomínky a pružný přístup k řešení problémů. Také bych rád poděkoval lektorům a konzultantům Jihomoravského inovačního centra za jejich pomoc s netechnickými aspekty našeho projektu.

Obsah

Kapitola 1

Úvod

Graffiti se stalo nedílnou součástí prakticky všech větších měst. Po dlouhou dobu bylo prakticky nemožné postihnout vandaly za škody jimi způsobené, neboť u nás není používána žádná databáze graffiti, která by umožnila tato data sbírat a analyzovat. Náklady na odstraňování graffiti se jen v České republice každoročně šplhají do desítek milionů korun. Objasněnost těchto činů je ale tristní. Tento stav dovedl postupně majitele nemovitostí až do jistého stádia apatie, kdy nové přírůstky často ani nehlásí, neboť šance na identifikaci pachatele je mizivá.

Tagů (zjednodušeně si je můžeme představit jako podpisy autorů) se v našich ulicích nachází obrovské množství a není jednoduché se v nich zorientovat. Pokud by však existoval systém, který by byl schopen tato data efektivně spravovat, mohli bychom tento jev detailně studovat a zefektivnit tak přijatá protiopatření. Navíc by bylo možné pachatele, jejichž dopadení je v současnosti především dílem náhody, přimět k nápravě způsobených škod.

Tato myšlenka se stala nosnou idejí projektu TagBust, který si vytvoření takového systému klade za cíl. S tímto projektem jsme úspěšně prošli akcelerátorem StarCube na Jihomoravském inovačním centru. V současné době je náš systém nasazen ve zkušebním provozu za účelem sběru dat městskou policií v Břeclavi.

Velkou motivací pro nás je fakt, že v zahraničí podobné systémy již úspěšně fungují. Zatím jsme ale neodhalili žádný, který by se hlásil k použití technologií počítačového vidění, což je ovšem pravděpodobně pouze otázkou času. Tyto systémy bývají rozšířeny o různé přenosné kamerové systémy nebo online fotopasti, čímž přispívají ke zvýšení úspěšnosti dopadení pachatelů. Důležitou motivací pro nás je skutečnost, že tyto systémy mají reálný pozitivní dopad na objasněnost a následné snížení výskytu tohoto druhu trestné činnosti. Systém G.R.I.P.[1] se chlubí snížením počtu nových graffiti o 50 % za první rok používání systému. Nasazení systému se městům vyplatí i z ekonomického hlediska, neboť se podařilo vymoci část škod způsobených vandaly.

Cílem této práce je tedy navrhnout a implementovat algoritmus, který by umožňoval v rámci takové databáze vyhledávat podobné podpisy, a tím by umožnil její efektivní správu. Díky svému charakteru se tato problematika podobá rozpoznávání podpisů a způsobem vyhledávání má jisté společné prvky s multimediálními databázemi, které umožňují mimo jiné i vyhledávání podle podobnosti obrázků.

Pod hlavičkou projektu TagBust jsou zpracovávány celkem tři diplomové práce. Vedoucí týmu Matěj Kubiš se ve své práci [12] zabývá aplikací pro sběr dat, která bude určena široké veřejnosti. Vojtěch Grünseisen pak zpracovává alternativní způsob přístupu k vyhledávání

[1] http://www.gripsystems.org/

v databázi pomocí algoritmů používaných k vyhledávání obrázků [8].

Kapitoly této práce jsou členěny tímto způsobem: následující kapitola se zabývá analýzou problému z pohledu důležitého pro návrh algoritmu, kapitola 3 popisuje vybrané přístupy a další algoritmy používané v souvislosti s rozpoznáváním podpisů. V kapitole 4 je diskutován návrh řešení, kapitola 5 popisuje implementaci tohoto řešení, kapitola 6 popisuje výsledky testování jednotlivých algoritmů a kapitola 7 obsahuje závěrečné shrnutí dosažených výsledků a nástin budoucího vývoje systému.

Kapitola 2

Analýza problému

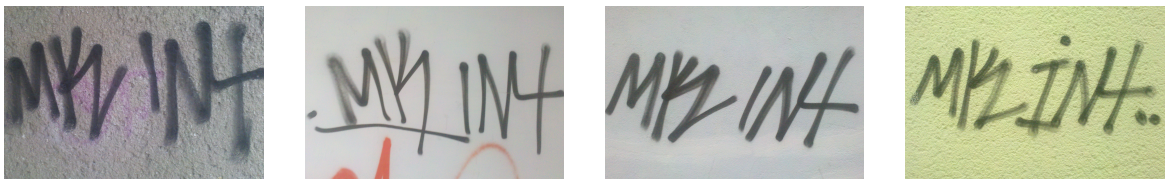
Graffiti subkultura se před veřejností uzavírá a její reálie nejsou obecně známé. Proto je pro začátek nutné definovat některé základní pojmy a popsat základní principy. Tato práce si však neklade za cíl graffiti zkoumat z jiného hlediska než čistě technického, proto se autor předem omlouvá za případné nepřesnosti a zjednodušení z toho plynoucí.

2.1 Graffiti tag

Počátek novodobého graffiti se datuje v 70. letech minulého století v New Yorku. Jednalo se o nápis *JULIO 204* sprejem na zdi domu. Tento nápis údajně objevil poštovní doručovatel, který zjistil, že číslo 204 značí ulici. Pochopil, že se jedná nejspíše o přezdívku autora a odpověděl v podobném duchu, tedy začal sprejovat svoji přezdívku doplněnou o číslo ulice – *TAKI 183*. A to v takovém rozsahu, že se obyvatelé New Yorku začali zajímat, o co se jedná. V České republice byly první výskyty graffiti zaznamenány krátce po Sametové revoluci.

Takto tedy vznikly první graffiti *tagy*. V podstatě se tedy jedná o podpis autora, který se v graffiti subkultuře nazývá *writer*. Již z legendy o vzniku graffiti plyne základní motiv jeho autorů, tedy šířit zprávu o svojí existenci.

Přestože *writer* čas od času svůj podpis změní, je možné říci, že tento slouží jako identifikátor autora. Pokud by *writer* měnil svůj podpis příliš často, přestal by naplňovat jednu ze základních motivací autora. Z toho plyne paralela mezi graffiti tagy a lidskými podpisy. Tag ale není jediným projevem graffiti, z průzkumu situace v Brně je jasné, že jde o projev

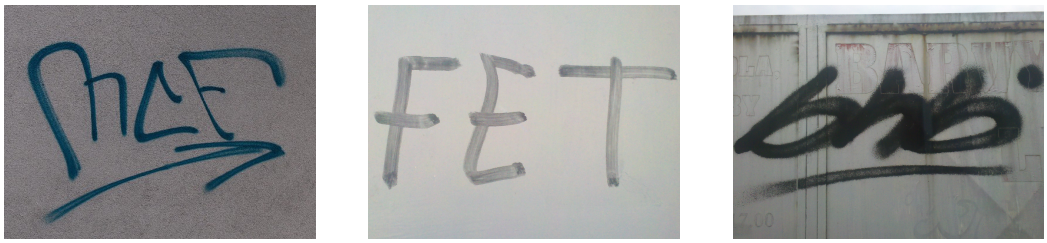


Obrázek 2.1: Několik variant jednoho tagu.

nejčastější. Vytvoření tagu zabere *writerovi* jen několik málo sekund. Podobně časově náročné je sprejování přes šablonu. Na místech, kde je to možné se objevují i komplikovanější graffiti. Jak je zřejmé z obrázků 2.2a a 2.2b, jejich vytvoření je již náročnější na čas i zdroje. Tato práce se však zabývá pouze klasifikací tagů, proto by byl rozbor různých dalších typů graffiti nad její rámec. Základní fakta a informace o graffiti v této sekci byla čerpána z [19].

(a) (b)

Obrázek 2.2: a) Legální graffiti; b)Nelegální graffiti.



(a) (b) (c)

Obrázek 2.3: a) MCF; b) FET; c) BHB.

2.2 Druhy tagů

Naopak je důležité rozlišovat tagy podle autorství. Autoři výše uvedených tagů se často sdružují do skupin, takzvaných *crew*. Jedná se o základní organizační jednotku graffiti subkultury. Jistým způsobem se *crew* podobá gangu. V České republice se význam slova *crew* asi více blíží slovu parta, neboť zde na rozdíl od USA nedochází k takovému propojení s organizovaným zločinem, který graffiti využívá pro např. pro označení svého území [2]. Z technického hlediska se jedná o skupinu lidí, kteří píší stejný tag. Jeho autorem tak není jen jeden člověk, ale celá skupina.

Tag *crew* bývá zkratkou jejího názvu a typicky se skládá z 2 až 4, nejčastěji pak 3 písmen [20]. Tagy několika brněnských *crew* jsou vidět na obrázcích **??** až **??**. Z analýzy tagů v Brně je možné říct, že tag *crew* bývá poměrně dobře čitelný. Oproti tomu tag *writera* nemívá konkrétní význam, důležitá je vizuální podoba a vyznění jména [20]. Oproti tagu *crew* se obvykle skládá z více písmen a může být komplikovanější, jak je vidět na obrázcích 2.3a až 2.3c. Ne vždy je proto pro laika čitelný. Z poslední statistiky plyne, že strážníci městské policie v Břeclavi byli schopni při testovacím provozu přepsat cca. 30% tagů.

2.3 Výskyty tagů

Poměrně důležitým faktorem je charakteristika rozmístění tagů. Protože záměrem *writera* je být co nejvíce vidět, vyskytují se tagy často na frekventovaných místech, jako jsou např. trasy tramvají. Čím rizikovější místo, ať už z hlediska fyzického nebezpečí nebo možnosti

(a) (b) (c)

Obrázek 2.4: a) Nečitelný tag; b) Emil; c) Nečitelný tag

odhalení, tím větší prestiž tag autorovi přinese.

Tagy vznikají často v nárazových akcích. Z informací od Městské policie Břeclav je znám případ, kdy jeden *writer* během několika hodin spotřeboval tři plechovky spreje a nasprejoval několik desítek tagů. Rovněž ze zkušeností autora plyne, že nové tagy ve sledovaných lokalitách přibývají ve vlnách a je možné vysledovat trasu, kudy se jejich autoři pohybovali. Dle informací od Policie dochází k dopadení zkušených sprejerů velmi zřídka, obvykle se jim podaří zadržet pouze začátečníky. Jedním z uvedených důvodů je i to, že zkušení writeři používají při akci hlídače, takzvané *checkery*.

Na základě těchto informací je možné předpokládat, že tento model vzniku tagů je obecně rozšířený. Tuto domněnku potvrzuje skutečnost, kdy se často stejné tagy vyskytují blízko sebe, což bylo zjištěno zkoumáním v terénu. Tento fakt je možné využít pro optimalizaci porovnávacího procesu, kdy je možné nejprve pro dotazovaný tag prohledávat jeho okolí.

Z tohoto důvodu může být čas vzniku (respektive pořízení fotografie) zajímavým vodítkem při porovnávání tagů a hledání souvislostí mezi nimi a jejich autory. Klade však vysoké požadavky na aktuálnost dat v systému.

2.4 Podobnost s podpisy

Jak již bylo řečeno, zadaná úloha je podobná rozpoznávání podpisů. Oproti tomu však skýtá určité problémy, které se při rozpoznávání podpisů nevyskytují. Tyto problémy jsou v následujících řádcích dále rozebrány. Naprostá většina offline metod pro rozpoznávání podpisů operuje s binárním obrazem podpisu odděleného od pozadí. V případě rozpoznávání podpisů je však segmentace zjednodušena nejen použitím bílého papíru jako podkladu, ale také předem danou pozicí a orientací snímaného podpisu. Z obrázku 2.4a je zřejmé, že rozpoznání správné orientace (respektive roviny) tagu může být v některých případech obtížné i pro člověka. Tento snímek taktéž demonstruje poměrně hojně se vyskytující problém s přílišnou komplexností pozadí, které má negativní vliv na úspěšnou segmentaci.

Nerovný povrch Z obrázku 2.4b je zřejmé, jaké komplikace se vyskytují v případě, že je tag nasprejován na členitém povrchu. Podobný efekt se dostaví i v případě tagu umístěném např. na cihlové zdi. Speciální komplikací v tomto případě může být barva spreje shodná s výplní mezi cihlami, např. zeď z pálených cihel se světlou výplní postříkaná sprejem světlé barvy. Do podobné kategorie lze zařadit i případ, kdy z nezaschlého tagu stéká barva a propojí tak linie tagu mezi sebou. Tento jev je možné pozorovat např. na již dříve uvedeném obrázku 2.3b.

Ostré stíny V případě vystavení snímaného tagu ostrému stínu dochází rovněž k situaci, kdy je obtížné tag úspěšně segmentovat. V případě tagu na obrázku 2.4c by nepomohlo ani použití blesku na běžném mobilním telefonu. Při pořizování dat je nutné brát v potaz vlastnosti použitého zařízení a zajistit dobré osvětlení pro získání dat v dostatečné kvalitě s konzistentním osvětlením.

Prolínání s pozadím Další komplikací je prolínání tagu s podkladem. K tomuto jevu dochází při v závislosti na použité trysce spreje a vzdálenosti trysky od zdi. Mezi linií tagu a podkladem bývá nejasná hranice, což komplikuje rozhodnutí, zda–li se ještě jedná o oblast tagu, nebo ne. Na obrázku 2.4d došlo zvětšením vzdálenosti od plochy k takovému stupni prolnutí s podkladem, že při segmentaci dojde pravděpodobně k přerušení samotné linie tagu. Tento problém je častý zejména u starších, již vybledlých tagů.

Překrývající se tagy Na obrázku 2.4e je téměř nemožné od sebe odlišit jednotlivé tagy. Ačkoliv se v tomto případě jedná o extrémní příklad, tento problém se vyskytuje relativně často, byť v nižší intenzitě. Pokud mají tagy stejnou barvu, bývá problém rozhodnout, zda se jedná o dvě části jednoho tagu, nebo dva zcela odlišné tagy. Někdy je toto rozhodnutí možné učinit až na základě získaných znalostí o obvyklé podobě daného tagu.

Z těchto příkladů plynou požadavky na kvalitu a anotaci dat a technická omezení systému jako takového. Následující soupis obsahuje jak tyto technické požadavky, tak i požadavky vzešlé ze spolupráce s MP Břeclav.

Požadavky na snímky

- Osa kamery musí být ortogonální vůči snímané ploše (toto je důležité mimo jiné i pro případný písmoznalecký posudek).
- Musí být zajištěno konzistentní osvětlení po celé oblasti tagu.
- Na fotce musejí být označeny pozice tagů.
- Je–li to možné, umístit rovinu tagu rovnoběžně s okraji snímku.
- Snímek musí být anotován GPS souřadnicemi.
- Pro účely ruční přípravy dat je vhodné anotovat tagy jejich přepisem.

Pro vyhledávání v databázi tedy budeme považovat za vstup výřez z fotografie obsahující jeden tag, který bude zarovnán s horním nebo dolním okrajem a nebude zasahovat až k okrajům, respektive je přesahovat. V opačném případě nebude možné zajistit správné výsledky vyhledávání.

Dále je zvažováno získání vzorku barvy tagu pomocí zamíření na jeho část pomocí zaměřovače v mobilním zařízení. Tento přístup je však zatím v experimentální fázi a proto není zatím jasné, zda bude při masovém sběru dat veřejností použitelný. Implementace sběru dat ze strany veřejnosti je dále rozebrána v [12].

(a)

(b)

(c)

(d)

(e)

Obrázek 2.5: a) Neznámá orientace tagu; b) Členitý podklad; c) Ostré stíny; d) Prolínání s pozadím; e) Překrytí tagů.

Kapitola 3

Algoritmy používané pro rozpoznávání podpisů

Rozpoznání podpisu spočívá v jeho přiřazení ke konkrétnímu autorovi. Druhým typem úlohy je verifikace, což je rozhodnutí o pravosti podpisu. Podle [14] je možno podvržené podpisy dále dělit na **náhodné**, které jsou vytvořeny bez znalosti podpisu nebo jména autora, **prosté**, vytvořené pouze se znalostí jména a **kvalifikované**. Ty jsou vytvořeny podle získaného podpisového vzoru. V současnosti je podle [5] největší potenciál v úlohách verifikace a autorizace např. v bankovním styku a podobných aplikacích.

Metody pro rozpoznávaní podpisů se dělí na dvě základní kategorie podle způsobu získávání dat: *online* a *offline*. *Online* metody pracují s daty získanými pomocí specializovaných digitalizačních zařízení, např. aktivních tabletů. Díky tomu je možné snímat podpis ve vyšší kvalitě a sledovat mimo jiné i údaje o rychlosti, přítlaku a rytmu psaní. Výstupem snímání je sekvence souřadnic bodů a informace o zvednutí nebo položení pera. To činí online metody robustními vůči pokusům o padělání, neboť napodobit rytmus psaní je obtížné. Vhodným užitím pro tyto metody je např. ověření totožnosti pro přihlášení do operačního systému nebo vstupu do střežené oblasti. [6]

U *offline* metod se digitalizace provádí např. pomocí scanneru. Jejím výstupem je matice bodů, kterou je nejprve nutné při předzpracování převést do vhodné reprezentace pro klasifikaci. *Offline* metody nejsou tak odolné vůči podvrhům, ze své podstaty ale nevyžadují speciální zařízení pro pořízení a mohou tak být využity např. ve forenzních aplikacích. [6]

Podle druhů extrahovaných příznaků je možné metody pro rozpoznávání podpisů rozdělit na tři základní kategorie [6]:

Holistická metoda pracuje s vektorem příznaků extrahovaných z podpisu, vektory jsou porovnávány na základě vzdálenosti. Jedná se o globální příznaky jako např. poměr šířky a výšky, těžiště podpisu, počet křížení a podobně.

Regionální metoda pracuje se sekvencí vektorů a porovnává vektory s ohledem na strukturu v sekvenci.

Lokální metoda pracuje s časovou a prostorovou aproximací dat a provádí elastické porovnání. Příkladem může být např. aproximování podpisu pomocí úseček [15].

3.1 Příznaky

Následuje popis některých používaných příznaků pro rozpoznávání podpisů.

Poměř šířky a výšky Fyzická velikost podpisu je proměnlivá, poměr stran obalujícího obdélníku však zůstává podobný. Poměr je vypočten podle následujícího vzorce: [16]

$$\begin{aligned} \gamma &= \frac{w}{h},\ pokud\ w \geq h \\ \gamma &= -\frac{h}{w},\ pokud\ w < h \end{aligned} \tag{3.1}$$

Horizontální a vertikální těžiště Souřadnice těžiště lze vypočíst pomocí rovnic **??** a **??** na skeletonizovaném obraze. [14]

$$Center_x = \frac{\sum_{x=1}^{x_{max}} x \sum_{y=1}^{y_{max}} b[x,y]}{\sum_{x=1}^{x_{max}} \sum_{y=1}^{y_{max}} b[x,y]} \tag{3.2}$$

$$Center_y = \frac{\sum_{x=1}^{x_{max}} y \sum_{y=1}^{y_{max}} b[x,y]}{\sum_{x=1}^{x_{max}} \sum_{y=1}^{y_{max}} b[x,y]} \tag{3.3}$$

Počet koncových bodů Za koncový bod je považován takový, který má ve svém osmiokolí pouze jednoho souseda. Vstupem pro extrakci tohoto příznaku je skeletonizovaný obraz. [11]

Počet křížení Vstupem je opět skeletonizovaný obraz. Za křížení je brán takový bod, který má ve svém osmiokolí alespoň tři sousedy. [11]

3.2 Vertikální a horizontální projekce

Tato metoda uvedená v [16] popisuje podpis pomocí jeho projekce na osy x a y, čímž je získán histogram hustoty pixelů. Histogram je získán dvěma průchody a jeho výsledkem jsou dvě jednodimenzionální pole (T_v pro vertikální a T_h pro horizontální směr). Po získání histogramu jsou histogramy převzorkovány na normalizovanou velikost odpovídající požadované velikosti obrazu, kde jedna strana má délku n.

Nejprve je získán normalizační koeficient δ podle vzorce **??**. Pomocí něho jsou následně vypočteny normalizovaná projekční pole N_v dle vzorce **??** a N_h podle **??**.

$$\delta = \frac{\max\{X, Y\}}{n} \tag{3.4}$$

$$N_v[i] = round(\frac{T_v[round(i * \delta)]}{\delta}), i = 0, ..., n-1 \tag{3.5}$$

$$N_h[i] = round(\frac{T_h[round(i * \delta)]}{\delta}), i = 0, ..., n-1 \tag{3.6}$$

Následné porovnání pak bere ohled na fakt, že dané projekce od sebe bývají posunuty, proto jsou zde porovnávány s odchylkou $\Delta d = \pm 10$ pixelů a výsledek je uložen do polí T_v a T_h. $N_{1,v}$,$N_{2,v}$,$N_{1,h}$ a $N_{2,h}$ jsou vertikální, respektive horizontální koeficienty podpisů S_1 a S_2.

$$T_v[i] = |N_{2,v}[i] - N_{1,v}[i]|, i = 0, ..., n-1 \tag{3.7}$$

$$T_h[i] = |N_{2,h}[i] - N_{1,h}[i]|, i = 0, ..., n-1 \tag{3.8}$$

Následně jsou spočteny parciální koeficienty podobností pro každou tabulku:

$$\sigma_v = \sum_{i=0}^{n-1}(1 - \frac{T_v[i]}{n}) \tag{3.9}$$

$$\sigma_h = \sum_{i=0}^{n-1}(1 - \frac{T_h[i]}{n}) \tag{3.10}$$

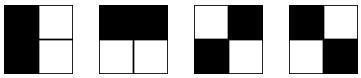
Výsledný koeficient podobnosti je pak vypočten na základě tohoto vzorce:

$$s_p = \frac{\max(\sigma_{-d,v}, ..., \sigma_{d,v}) + \max(\sigma_{-d,h}, ..., \sigma_{d,h})}{2} \tag{3.11}$$

Tato metoda pro porovnání dvou podpisů byla převzata z [16].

3.3 Porovnávání podpisů pomocí strokes

Na rozdíl od ostatních příznaků se v tomto případě jedná o příznaky lokální, jejichž způsob vyhodnocování je svým způsobem specifický. Vstupem tohoto algoritmu je skeletonizovaný obraz. Ten je postupně skenován pomocí čtyřech různých kernelů viz obrázek 3.1. Tímto způsobem jsou označeny pixely podle příslušnosti k orientovaným hranám.



Obrázek 3.1: Kernely pro detekci hran.

Tabulka 3.1: Pořadí průchodu pixely při spojování orientovaných hran.

Úhel	Pořadí průchodu
0°(Z)	{Z, JZ, SZ, S, J, SV, JV, V}
0°(V)	{V, SV, JV, S, J, JZ, SZ, Z}
45°	{JZ, Z, J, JV, SZ, V, S}
90°	{J, JV, JZ, V, Z, SV, SZ, S}
135°	{JV, V, J, JZ, SV, Z, S}

Každá z těchto čtyř skupin je pak postupně procházena podle daného pořadí viz tabulka 3.1, čímž jsou postupně získány orientované úsečky daného směru, takzvané *stroky*. Pokud při procházení dojde algoritmus na pixel, z něhož nelze postupovat dále, pokouší se hledat, zda–li v blízkém okolí úsečka nepokračuje. Tím je zajištěno pokračování v případě výpadku linie v důsledku šumu, případně na křížení. Podobný přístup ale s využitím Houghovy transformace je popsán v [16].

Následně jsou odfiltrovány krátké stroky a zbylé jsou normalizovány. Výsledná podobnost S se vypočítá ze vzorce 3.7, kde $mat_{t,j}$ je počet shodných stroků z t nalezených v j a s_t je celkový počet stroků v t.

$$S_{t,j} = \min(\frac{mat_{t,j}}{s_t} \times \frac{mat_{j,t}}{s_j}) \tag{3.12}$$

Předmětem trénovací fáze je v této metodě nalezení prahových hodnot pro porovnávání koncových bodů stroků a jejich minimální přípustné velikosti. Algoritmus je převzat z [15].

3.4 Segmentace obrazu s použitím barevných a texturních příznaků

Tento přístup rozšiřuje algoritmus meanshift [4], který provádí segmentaci na základě barevných a prostorových informací o texturní příznaky. K charakteristice texturních příznaků jsou využity *diskrétní vlnkové rámace* [18].

1. Nejprve je obraz pomocí vlnkové transformace dekomponován do čtyřech pásem: DD, HH, HD a DH. Písmena D a H odpovídají tomu, zda se jedná a dolní (D), respektive horní(H) propust. Obraz je filtrován nejprve po řádcích, následně po sloupcích. Nejvíce texturních informací je obsaženo v pásmech HD a DH.

2. Energie je získána mediánovou filtrací koeficientů pásem HD a DH v rámci lokálního okna. Velikost okna musí být taková, aby zachytila charakteristiku textury. Energie je definována jako druhá mocnina koeficientu.

3. Podle orientace textury klasifikujeme pixel do jedné ze čtyř kategorií: vertikální, horizontální, nerozlišitelný (v žádné orientaci není dostatečná energie) a komplexní (žádná orientace nepřevládá). Nejprve je energie v HD a DH pásmu klasifikována do kategorií 0 a 1 s použitím algoritmu k-means [9]. Následně je provedena klasifikace do jedné ze čtyř uvedených kategorií: pixel je klasifikován jako vertikální, pokud má energii v pásmu DH rovnu nule a energii v pásmu HD rovnu jedné. V opačném případě by byl klasifikován jako horizontální. Pokud je v obou pásmech energie rovna nule, pak je pixel klasifikován jako nerozlišitelný, v opačném případě jako komplexní.

4. Následně je vygenerován vektor příznaků tak, že každý pixel má p–dimenzionální vektor, který obsahuje prostorovou (x,y), barevnou (šedotónovou nebo L*u*v) a texturní (vertikální, horizontální atd.) složku.

5. Obraz je filtrován pomocí algoritmu meanshift ve vytvořeném příznakovém prostoru. Výsledný obraz může být segmentován pomocí K-means. Filtraci lze ovlivnit nastavením parametrů.

Tato sekce byla čerpána z [13].

3.5 Skeletonizace

Proces, který vede k získání topologické kostry objektu se nazývá skeletonizace. Jedná se o morfologickou operaci. Postup skeletonizace je dán následujícím algoritmem [17]:

Skeleton $S(A)$ množiny A je dán jako:

- Pokud z je bod z $S(A)$ a $(D)_z$ je největší kruh se středem z obsažený v A, nelze nalézt větší kruh obsahující $(D)_z$ obsažený v A. D_z se nazývá maximální kruh.

- Kruh $(D)_z$ se dotýká okraje A na dvou nebo více různých místech.

Skelet lze definovat jako sjednocení středů maximálních kruhů:

$$S(A) = \bigcup_{k=0}^{K} S_K(A) \tag{3.13}$$

$$S_K(A) = (A \ominus kB) - (A \ominus kB) \circ B \tag{3.14}$$

Vpisování kruhů se ale prakticky nepoužívá kvůli velké výpočetní složitosti a porušení souvislosti linií. Z tohoto důvodu se používá výpočet skeletu pomocí **vzdálenostní transformace** [10]:

- Mějme dánu bodovou množinu A.
- Vzdálenostní transformací je přiřazena každému bodu $p \in X$ jeho vzdálenost od pozadí A^c (hranice oblasti).
- Morfologickým postupem výpočtu vzdálenostní transformace $dist_X(p)$ přiřazuje každému p z X velikost první eroze množiny, která neobsahuje p:

$$\forall p \in X, dist_X(p) = \min\{n \in \mathbb{N}, p \notin (X \ominus nB)\} \tag{3.15}$$

3.6 Vyhledávání v databázi obrázků

V dnešní době získává toto téma na významu, neboť je svět takřka zaplaven obrazovými daty, které je nutné třídit. Především se jedná o data bez anotací, ve kterých by jiným způsobem vyhledávat nebylo možné. Souhrnně téma se nazývá *Content Based Image Retrieval*, tedy CBIR. Tato funkcionalita je nedílnou součástí multimediálních databází jako je např. Oracle interMedia [1]. Primárním účelem je tedy nabídnout možnost třídit a vyhledávat v takové databázi podle charakteristik obrazu. Podle [7] je možné data řadit podle vzdálenosti mezi nimi. Ta může být stanovena na základě:

- **barvy**,
- **texturních příznaků** nebo
- **tvaru**.

Vyhledávání obvykle probíhá metodou *Query by Example*, kdy je na vstup vyhledávače zadán hledaný obraz a výstupem je množina vrácených výsledků na základě výše uvedených způsobů porovnávání.

[1] http://docs.oracle.com/cd/B19306_01/appdev.102/b14302/ch_cbr.htm

Kapitola 4

Návrh řešení

4.1 Sběr dat

Sběr dat pro ostré nasazení systému TagBust má na starosti kolega Matěj Kubiš, viz [12]. Z tohoto důvodu je zde rozebrán pouze návrh metody pro hromadný sběr dat, který poslouží k získání podkladů pro vývoj klasifikátoru [1].

Alternativně by tento přístup mohl být využit v případě, kdy by bylo třeba zmapovat větší území naráz (např. zmapování situace ve městě před zahájením opatření proti graffiti).

Fotografování Pro fotografování je možné využít prakticky libovolný fotoaparát. Je nutné zohlednit počet snímků, které lze nasnímat na jednu baterii a mít případně připraven záložní zdroj. Toto samozřejmě platí i pro paměťové médium.

Z ergonomického hlediska je vhodné využít kompaktní fotoaparát. Pokud se v mapované oblasti nachází např. podchody, je vhodné mít připraven blesk. Po provedených pokusech se ukázalo, že se pro tento druh snímání nehodí mobilní telefon, neboť dlouhodobější manipulace s ním není příliš uživatelsky přívětivá a oproti kompaktnímu fotoaparátu vyžaduje zbytečnou pozornost.

Geolokace Původním návrh počítal s využitím GPS přijímače v mobilním telefonu a zápisem tagu s pozicí do EXIFu. Tento přístup se však ukázal jako nevhodný, neboť použitý telefon (Motorola DEFY+) nedisponuje příliš kvalitním přijímačem a mezi budovami docházelo k častým ztrátám signálu, následkem čehož došlo v cca 1/3 případů k neotagování snímku. Takto získaná data jsou dále nevyužitelná.

Řešením se ukázalo být použití citlivého GPS přijímače. Ten snímá prošlou trasu a snímky jsou o GPS souřadnice doplněny dodatečně. Nutným předpokladem je synchronizace času ve fotoaparátu a GPS přijímači.

Následné zpracování a anotace Následné označení fotografií může být provedeno např. pomocí aplikace EasyGPS, která je dostupná zdarma. Jako vstup je použit záznam prošlé trasy z GPS přijímače. Pozice mezi uloženými *waypointy* jsou interpolovány na základě času. Následně jsou fotografie nahrány na server, kde jsou z každé fotografie extrahována EXIF data, která jsou uložena jako záznamy do MySQL databáze.

[1] Klasifikátor, ačkoliv je obvykle chápán jako program řadící jednotlivé vstupy do tříd, bude v této práci chápán volněji, tedy jako program porovnávající rozdíl mezi tagy.

Po tomto kroku je nutné provést dodatečnou anotaci, především tedy označit na fotce veškeré tagy a je–li to možné, uložit do databáze jejich přepis, případně označit tag jako nečitelný. Tuto anotaci je možné využít nejen při prvotním třídění dat pro manuální přiřazení do tříd v počáteční fázi, ale také pro vyhledávání konkrétních tagů v databázi.

4.2 Předzpracování

Vstupem *offline* metod pro rozpoznávání podpisů bývá binární obraz o normalizované velikosti, který je tzv. *skeletonizovaný*. To znamená, že tloušťka linie podpisu je pomocí algoritmu ztečena na šířku jedna. Tento fakt může být využit pro detekci příznaků, jako jsou např.:

- pozice křížení linií,
- koncové body,
- nebo extrahované linie (tzv. *strokes*).

Nejen v případě graffiti se však tímto způsobem ztrácejí určité kaligrafické prvky snímaného podpisu nebo tagu. V tomto případě se jedná např. o tloušťku linie. Ta se liší v závislosti na použité technice. Z tohoto důvodu budou jako základ pro další zpracování uchovávány binarizované snímky tagů.

Vstupem této fáze bude v praxi barevný výřez z původního snímku, který je získán prostřednictvím mobilní aplikace. Z něho bude následně extrahován tag. Pro účely vyhodnocování úspěšnosti segmentace je průběžně připravován set snímků, který bude tvořen dvojicemi snímků doplněných o jejich anotaci. Každá dvojice se bude skládat z původního výřezu z barevného snímku a ručně vytvořeného binárního obrazu tagu, se kterým bude porovnáván výstup segmentace. Tento set bude využit nejen pro případné automatizované ladění parametrů segmentace, ale i pro počáteční testování klasifikátoru. Dvojice snímků doplněná o skeletonizovaný obraz je vidět na obrázku **??**.

(a) (b) (c)

Obrázek 4.1: a) Původní výřez; b) Binarizovaný výřez; c) Po skeletonizaci.

Posledním krokem předzpracování je normalizace velikosti výřezu. Všechny binarizované výřezy jsou před dalším zpracováním normalizovány na velikost 512×512 pixelů, kdy se výřez normalizuje na větší rozměr z dvojice (výška, šířka). Následně je možné obraz skeletonizovat nebo odeslat na vstup extraktoru příznaků.

4.3 Extrakce popředí

Kvalitní segmentace popředí je pro další zpracování naprosto esenciálním krokem, neboť pouze tak je možné extrahovat relevantní příznaky bez nežádoucích artefaktů. Pro experimentování s různými triviálními metodami pro detekci popředí byl využit program GIMP. Pokusy však vyšlo najevo, že daný problém bude muset být vyřešen komplexnějším algoritmem, tedy propojit segmentaci založenou na barvě s lokací a charakterem linií. Na obrázku 4.2 je patrné, jak může špatná segmentace negativně ovlivnit další zpracování, např. skeletonizaci.

Obrázek 4.2: Žlutou barvou je vyznačen binarizovaný obraz tagu, černá linie značí výsledek skeletonizace.

Segmentace podle barvy, texturních informací a textury Pro neasistovanou segmentaci by mohl být využit algoritmus využívající prostorové a barevné příznaky doplněné o texturní informaci. Pro tuto metodu je stěžejní správné nastavení velikosti okna. Ta musí odpovídat šířce písma na podpisu. Parametry (jako např. váhy jednotlivých dimenzí nebo barevná vzdálenost) budou laděny pomocí automatického trénovacího procesu pomocí ručně segmentované trénovací sady.

SIOX Tento algoritmus je implementovaný v programu GIMP[2] jako nástroj pro segmentaci popředí. Je založen na hrubé selekci popředí pomocí *ROI* (tzv. Rectangle Of Interest) a následném označení vzorku popředí, na základě čehož je následně provedena segmentace. Jeho zásadním nedostatkem je však nutnost interakce s uživatelem, který musí označit vzorek popředí. Tento přístup není příliš efektivní z hlediska plánovaného způsobu nasazení systému, zároveň se jedná o výpočetně náročný algoritmus. Byl vybrán jako záložní varianta pro případ, že by se segmentaci nepodařilo vyřešit neasistovanou metodou. Tento algoritmus by však musel být implementován přímo na snímacím zařízení, protože by vyžadoval přímou interakci uživatele, proto zde nebude dále rozebírán. V případě jeho nasazení by byl jako vstup do klasifikátoru brán přímo segmentovaný obraz. Další informace o jeho implementaci lze nalézt v [1].

[2] http://www.gimp.org

4.4 Způsob použití v systému

V obvyklém systému pro identifikaci podle podpisu probíhá rozpoznání v rámci uzavřeného okruhu uživatelů. Od každého uživatele je na začátku získán podpisový vzor, zpravidla v počtu několika exemplářů, které slouží k natrénování klasifikátoru. Takový přístup je však pro graffiti tagy neaplikovatelný, neboť do systému budou průběžně přibývat nové třídy tagů. Zároveň zpočátku není možné počítat s výskytem dostatečného množství vzorků jednotlivých tagů v databázi pro počáteční trénink.

Zároveň by nebylo možné v systému o tomto plánovaném rozsahu pokrytí znovu trénovat klasifikátor při každém přidání nové třídy tagů. Problém představují i požadovaná množství tagů pro trénování, neboť během vkládání do systému nebude mít uživatel k dispozici dostatečné množství exemplářů jedné třídy pro trénink. Systém jako takový proto musí v prvé řadě umožňovat vyhledání tagů podle podobnosti (a dalších kritérií). To je také hlavní případ užití vybudované databáze – v případě dopadení sprejera při činu mít možnost provést vyhledání jeho díla v databázi, a tím předběžně zjistit rozsah jím způsobených škod. Následně může být osloven znalec z oboru písmoznalectví a vyhodnotit, zda jsou nalezené tagy skutečně dílem jednoho člověka.

Ze vstupního obrázku jsou extrahovány příznaky a následně je prohledána databáze, z níž je vráceno n nejpodobnějších výsledků. Při vyhledávání bude v praxi vzata v úvahu poloha tagu – vyhledávání bude probíhat od místa nálezu nového tagu podle vzdálenosti, dokud nebude nalezen určitý počet dostatečně podobných tagů, nebo překročen limit vzdálenosti pro vyhledávání. Tento způsob vyhledávání je podobný způsobům používaným v multimediálních databázích.

Samotné přiřazení do třídy je realizováno při dopadení nebo průběžně obsluhou systému (případně existuje varianta zpřístupnění této možnosti přímo uživatelům), která tímto způsobem postupně buduje databázi vztahů mezi jednotlivými tagy.

4.5 Vybrané metody pro porovnávání

Za účelem provedení **hrubého porovnání** z důvodu úsporu výkonu bylo navrženo prozkoumat porovnávání pomocí těchto příznaků:

- **Poměr stran** – jedno číslo určující poměr stran šířka × délka.
- **Těžiště** – souřadnice středu, tedy dvojice čísel.
- **Počet koncových bodů** – příznak umožňující odhad komplexnosti tagu.
- **Počet křížení** – podobně jako předchozí umožňuje třídit tagy podle komplexnosti.
- **Počet horizontálních a vertikálních maxim** – počet maxim na projekčním histogramu,
- **Počty stroků jednotlivých orientací** – čtveřice čísel vyjadřující počet stroků jednotlivých orientací v níže uvedené metodě pro porovnávání pomocí strokes.

Po počátečním odfiltrování zjevně odlišných tagů budou zbývající podrobeny důkladnějšímu porovnání pomocí těchto **komplexnějších metod**:

- **Porovnání pomocí strokes** – podpis je zjednodušen na množinu úseček a je provedeno porovnání, zda se úsečky nacházející v prvním podpisu nacházejí i v druhém a opačně.

- **Porovnání podle projekce** – porovnání průběhu podpisu z hlediska hustoty pixelů s určitou tolerancí oproti posunu.

Pro implementaci byl vybrán jazyk C++ v prostředí Code::Blocks. Primárně využívanou knihovnou bude OpenCV. Do budoucna je možný provoz na serveru, ten by tedy měl umožňovat spouštění C++ aplikací. Z cloudových služeb toto splňuje např. Microsoft Azure.

Kapitola 5

Implementace

V této kapitole je rozebrán používaný framework, implementace jednotlivých klasifikátorů a taktéž jednotlivé použité skripty. Programová část byla zpracována v prostředí MS Windows. K vývoji bylo využito prostředí `Code::Blocks`[1] s kompilátorem `MinGW`. Program je napsán v jazyce `C++`, nejvíce využívanou knihovnou je `OpenCV`[2] ve verzi `2.4.2`. Pro skriptování byl využit jazyk `Python`[3] verze `2.7`.

5.1 Framework

V počátcích projektu spolupracoval na vývoji klasifikátorů celý tým. Během této spolupráce byl navržen základní framework pro testování různých klasifikátorů. Dále bylo vytvořeno několik modulů, které jsou v práci dále použity k různým dílčím úlohám.

Celý framework je navržen tak, aby bylo možno nezávisle vyvíjet a používat jednotlivé klasifikátory. Každý klasifikátor musí implementovat definované rozhraní a je zkompilován jako dynamická knihovna. Takto je možné jednotlivé knihovny načítat dynamicky za běhu aplikace, což je výhodné pro testování a porovnávání výsledků jednotlivých klasifikátorů. Povinně definované metody jsou tyto:

`int init(cv::Mat image,cv::Mat origImage)` inicializuje instanci klasifikátoru na základě vstupních parametrů, kde první z nich je matice obsahující skeletonizovaný obraz a druhý obsahuje pouze segmentovaný obraz. Obě tyto matice mají pevně daný rozměr, a to 512×512 pixelů. Druhý z parametrů byl přidán až později. Dle posledních poznatků by bylo vhodnější mít pouze jednu vstupní matici a skeletonizaci provádět za běhu, nicméně prvotní implementace počítala se skeletonizovaným obrazem jako defaultním vstupem, až následně bylo zjištěno, že je pro určité implementace nutné mít i původní předzpracovaný obraz. Tento stav je tedy zachován kvůli zpětné kompatibilitě s již hotovými klasifikátory.

`int saveToFile(std::string path)` uloží inicializovaný klasifikátor do souboru, jako parametr je název souboru s klasifikátorem. Každému z klasifikátorů je přidělen suffix, podle kterého je možné identifikovat, kterému klasifikátoru soubor náleží. Soubor se ukládá ve formátu `YAML v. 1.0`[4]. K jeho hlavním přednostem patří zejména fakt, že

[1] http://www.codeblocks.org/
[2] http://opencv.org
[3] http://www.python.org
[4] http://www.yaml.org

je podporován knihovnou OpenCV, která obsahuje obslužné funkce a umožňuje tak jednoduše ukládat datové typy jí používané do souboru.

`double compareWithData(std::string path);` provede porovnání tagu, kterým je inicializována s tagem načteným ze souboru. Obvykle vrací skóre vzájemné podobnosti porovnávaných tagů v intervalu $\langle 0.0, 1.0 \rangle$. V případě některých klasifikátorů však toto nebylo dodrženo z důvodu odlišného hodnocení výsledků. V těchto případech je nutné zvlášť vyřešit zařazení do hodnotícího mechanismu.

`double compareWithImage(std::string path);` je používána k podobnému účelu jako předchozí metoda, ale nenačítá tato data z vybudované databáze, ale provede inicializaci přímo ze zadaného snímku. Není běžně používána z výkonnostních důvodů, u některých klasifikátorů již ani není implementována.

Protože v původním návrhu nebylo počítáno se spojováním výsledků z více klasifikátorů, bylo nutné provést za tímto účelem různě obsáhlé úpravy. Původním autorem návrhu a implementace je Matěj Kubiš.

Použití frameworku Framework je ovládán pomocí parametrů příkazové řádky. Kromě definice rozhraní klasifikátorů jeho základní funkcionalita umožňuje:

- **vybudovat databázi**,
- **vizualizovat výsledky vyhledávání do souboru**,
- **spouštět vyhledávání v tazích uložených v databázi a**
- **spouštět vyhledávání v obrázcích tagů.**

Protože se jedná o experimentální implementaci, byly rutiny obsluhující vyhledávání v databázi, testování a ladění parametrů zpracovány v jazyce Python s využitím funkcionality tohoto frameworku.

Výstavba databáze Databází pro účely této práce rozumíme soubory s extrahovanými informacemi ze vstupních dat. Nejprve jsou načteny dynamické knihovny s klasifikátory ze složky `classify/bin/classifiers` a je ověřeno, zda implementují všechny všechny rozhraním definované metody.

Databáze je následně budována nad obrázky ze složky `classify/data`. Pro každý z nalezených klasifikátorů postupně prochází tuto složku, načítá předzpracované tagy a k nim náležící skeletony ze složky `classify/data/skelets` a touto dvojicí klasifikátor inicializuje. Následně s využitím metody klasifikátoru zapíše reprezentaci tagu pro daný klasifikátor do souboru.

Název tohoto souboru je tvořen názvem původního souboru s předzpracovaným obrázkem a suffixem, který je každému klasifikátoru přidělen. Tímto způsobem je možné rozlišit soubory náležící jednotlivým klasifikátorům. V rámci této práce byly vytvořeny tři klasifikátory pracující s vlastními daty, jimž přidělené suffixy je možné nalézt v tabulce 5.1. Soubor kromě užitečných dat vždy obsahuje datum vytvoření a identifikaci příslušného klasifikátoru.

Tabulka 5.1: Přiřazení suffixů ke klasifikátorům.

Klasifikátor	Přiřazený suffix
StrokeComparator	`s`
Projections	`p`
AspectRatio	`r`

Proces porovnávání Porovnávací proces probíhá obdobně jako proces budování databáze. Inicializace instance však pro každý z klasifikátorů probíhá pouze jednou. Poté je provedeno porovnání oproti celé databázi tagů, respektive přímo oproti segmentovaným snímkům tagů a výsledky jsou uchovávány v datové struktuře `std::list`. Při ukládání do této struktury dochází zároveň k násobení výsledků jednotlivých klasifikátorů s jejich vahami.

Po provedení všech porovnání jsou výsledky seřazeny a vypsány na výstup, aby mohly být dále zpracovány prostřednictvím skriptů. Framework taktéž provádí základní vizualizaci výsledků, kdy do souboru `classify/data/results/single_result.html` zaznamená výsledky aktuálního testu. A to jak celkový výsledek, tak i dílčí výsledky jednotlivých klasifikátorů.

5.2 Skeletonizace

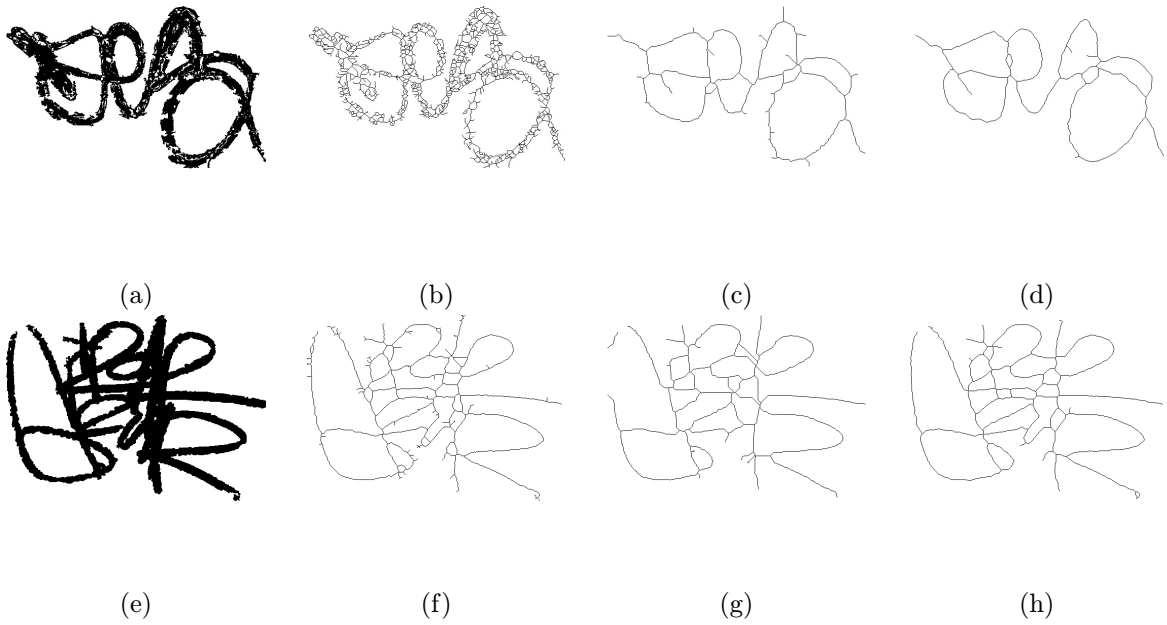
Algoritmy pro skeletonizaci byly pouze otestovány, neboť až na dílčí úpravy byla jejich implementace převzata. Testování těchto algoritmů potvrdilo důležitý rozdíl mezi graffiti tagy a podpisy. Zatímco u podpisů je tloušťka čar vesměs konstantní, pro tag je to jedna z důležitých charakteristik, které jsou specifické pro různé autory. Tloušťka a průběh linie závisí na typu použité trysky, rychlosti tahu, úhlu a vzdálenosti ústí trysky od podkladové plochy. Změnou těchto parametrů je možné dosáhnout výsledku na fotografii **??**. Zde zachycený tag má široké splývající linie, ve kterých navíc výrazně prosvítá podkladová plocha. V tomto případě při segmentaci (viz **??**) došlo k lehké fragmentaci a sloučení linii, což v závěru vedlo k deformaci výsledného skeletonu (viz **??**).

(a) (b) (c)

Obrázek 5.1: a) Původní snímek; b) Po segmentaci; c) Skeletonizovaný tag.

Nejdůležitějším kritériem pro vyhodnocení výsledků je množství artefaktů ve výsledném obraze a požadavek, aby skeleton co nejvíce reprezentoval tvar původního segmentovaného obrazu. Z hlediska vhodnosti byly otestovány následující tři implementace:

- Algoritmus **Guo-Hall** [5] je implementačně jednoduchý, ale neposkytuje prakticky žádnou ochranu proti vzniku artefaktů.
- Algoritmus **Zhang-Suen**[6] Z testovaných algoritmů nabízí vyvážený poměr mezi množstvím artefaktů a časovou náročností, z tohoto důvodu je dále používán.
- Algoritmus **Chatbri-Kameyama**[7] se ukázal být nejvíc odolným vůči špatně segmentovaným datům, je to ale bohužel vykoupeno jeho časovou náročností, kdy zpracování jednoho tagu trvá řádově jednotky minut i přes relativně malou velikost obrázků (512×512 pixelů). Implementace je popsána v článku [3].



Obrázek 5.2: a), e) Segmentovaný obrázek; b), f) alg. Guo-Hall; c), g) alg. Zhang-Suen; d), g) alg. Chatbri-Kameyama.

5.3 Proces přidávání dat

Ačkoliv bylo původně v plánu vyvinutí automatického algoritmu pro extrakci tagů ze snímku, bylo nakonec od tohoto záměru upuštěno, neboť v této fázi projektu není tato funkcionalita potřebná. Zároveň se jedná o dost komplexnější problém, než bylo původně předpokládáno.

Vstupní data byla průběžně vytvářena s pomocí volně dostupného programu GIMP. Výstupem této fáze jsou snímky tagů jako binární obraz, kde popředí je tvořeno nulovými hodnotami, pozadí hodnotami maximálními (255). Ačkoliv by bylo logické obrácené uspořádání, bylo toto schéma zvoleno s ohledem na obvyklé používání barev, kde bílá bývá

[5] Zdroj implementace:
http://opencv-code.com/quick-tips/implementation-of-guo-hall-thinning-algorithm/
[6] Zdroj implementace:
http://permalink.gmane.org/gmane.comp.lib.opencv/36588
[7] Zdroj implementace:
http://adapt.cs.tsukuba.ac.jp/~chatbri/web/publications.html

standardně považována za pozadí. Data jsou ukládána pomocí bezeztrátové PNG komprese, která je pro zaznamenání tohoto druhu dat vhodná. Naopak není vhodné používat ztrátové algoritmy jako např. JPEG, který vytváří charakteristické artefakty.

Následně je nutné snímky škálovat na unifikovanou velikost, která byla stanovena na 512×512 pixelů. Ze snímku s tagem jsou oříznuty nevyužité okraje a tag je škálován tak, aby byl jeho delší rozměr roven 512 pixelům a je umístěn do levého horního rohu. Při normalizaci velikosti je použita metoda interpolace *nearest neighbor*, při které je pixelu v novém obrazu přiřazena hodnota nejbližšího pixelu v původním obraze. Takto je zajištěno, že se na snímku vyskytují pouze hodnoty 0 a 255.

Posledním krokem pro rozpoznávání je spuštění skeletonizačního algoritmu uvedeného v předchozí sekci. Ten pro snímky vytvoří jejich skeletony. Soubory se skeletony obsahují v názvu suffix `_s`. Průběh tohoto procesu je zobrazen na obrázcích **??** a 4.1c.

Pro zjednodušení tohoto procesu byl za tímto účelem vytvořen skript v jazyce Python, který tento proces provede automatizovaně. Tento skript se spolu s návodem k použití nachází ve složce `data_script` na přiloženém DVD.

Takto připravený dataset postačuje k provádění jednotlivých experimentů. Pro automatické vyhodnocování výsledků je však nutné mít pro každý dotaz seznam očekávaných výsledků. Z důvodu jednoduchosti byla pro tento účel navržená struktura využívající objekt jazyka Python–dictionary. Formát souboru je následující:

```
{
...
"cts": ["CTS_1_1.jpg", "CTS_1_3.jpg", ..., "CTS_1_7.jpg"],
"mcf_arrowrl": ["2012-08-02 15.48.07_0.png", "MCF_5_28.jpg", "MCF_5_3.jpg"]
....
}
```

Jako klíč ve slovníku je použit textový identifikátor třídy, nejčastěji se jedná o její přepis, je–li tag čitelný. Jako hodnota přiřazená ke klíči je seznam všech obrázků obsahujících tag stejné třídy. Tento slovník je uložen do souboru pomocí serializéru JSON[8]. Tento soubor lze vytvořit s použitím skriptu `create_classes.py`, který je umístěn ve složce `classify/bin/create_classes.py`. Ten projde složku `data/classes`, ve které se nacházejí jednotlivé složky se snímky, které patří do jedné testovací třídy. Název složky je použit jako identifikátor třídy a jména jednotlivých souborů jsou uložena jako identifikátory tagů.

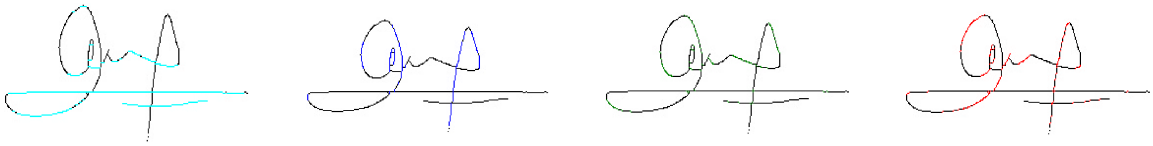
5.4 Porovnávání pomocí strokes

Inicializační fázi komparátoru je možné rozdělit na fáze

- **skenování**,
- **filtrace** a
- **normalizace**.

[8] http://www.json.org

Skenovací fáze Fáze skenování je implementačně nejobsáhlejší. Vstupem je skeletonizovaný obraz, na kterém je v první fázi provedena filtrace pomocí filtru pro detekci orientovaných hran o rozměrech 2×2 pixelů. Filtrace je prováděna pro 4 uvažované směry – horizontální, vertikální, 45°a 135°. Pro každou orientaci je filtrace provedena dvakrát, kdy jsou jako oba možné body kernelu (viz 3.1) uvažovány jako kotevní a oba výsledné obrazy jsou následně sloučeny pomocí bitové operace nad poli. Výsledkem tedy jsou 4 matice pixelů, které obsahují pixely označené orientací příslušné hrany. Platí, že každý pixel se může vyskytovat ve více hranách. Výsledky tohoto skenování jsou vyobrazeny na obr. 5.2a až 5.2d.



(a) (b) (c) (d)

Obrázek 5.3: Pixely označené podle náležitosti ke hranám: a) vodorovné hrany; b) vertikální hrany; c),d) úhlopříčné hrany. Obrázek převzat z [15].

Pro každou z těchto matic je následně provedeno skenování orientovaných hran. Při něm je obraz procházen po řádcích (s výjimkou skenování horizontálních hran) a z každého nalezeného pixelu je prováděno expanze směrem, který odpovídá příslušné orientaci podle tabulky 3.1.

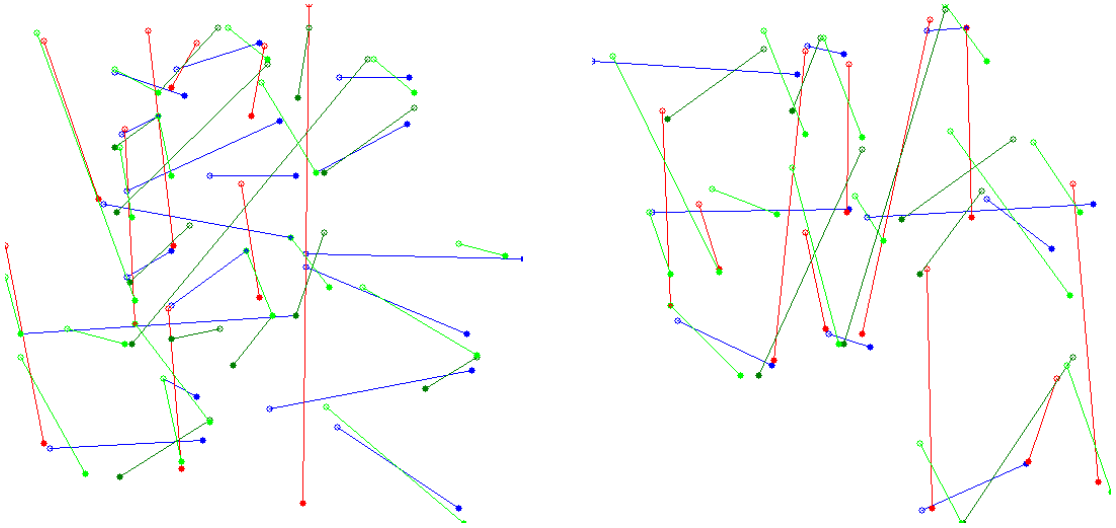
(a) (b) (c)

Obrázek 5.4: a) První nalezený pixel při horizontálním procházení; b) expanze z tohoto pixelu; c) rozdělený stroke.

Výjimka ve způsobu procházení pro horizontální směr je dána faktem, že každý zpracovaný pixel je označen a není možné ho dále procházet. Při skenování horizontálním kernelem po řádcích zleva doprava by tedy došlo ke stavu, kdy by prvním expandovaným pixelem byl žlutě zvýrazněný pixel na obr. 5.3a, následně by byly naskenovány a označeny pixely ve směru skenování, tedy vpravo od pixelu viz obr. 5.3b. Začátek linie by byl naskenován až při procházení dalšího řádku, čímž byl tento stroke rozdělen na dva viz obr. 5.3c, což je nežádoucí stav.

Tento způsob detekce orientovaných hran je poněkud nespolehlivý a proto je nutné při nalezení koncového bodu zkontrolovat, zda stroke nepokračuje po výpadku dále. Vzdálenost, kterou je možno přeskočit, pak byla předmětem testování a nakonec se ustálila na 15 pixelech. Ladění tohoto parametru je poměrně obtížné, neboť je v každé iteraci nutné vždy znovu vybudovat databázi pro daný klasifikátor. Ladění tohoto parametru zatím není pokryto funkcionalitou skriptu.

V případě, že je dosaženo bodu, ze kterého nelze pokračovat dále a ve vzdálenosti, kterou je možno přeskočit se již žádný pixel náležící ke stejně orientované hraně nenachází, je daný bod považován za koncový bod daného stroku. Výsledkem vyhledávací iterace je tedy trojice



(a) (b)

Obrázek 5.5: Výsledky inicializačního procesu.

tvořená počátečním bodem, koncovým bodem a orientací daného stroku. Výstupem celého procesu je množina těchto trojic.

Filtrace a normalizace Výstupem předchozí fáze je množina úseček s přiřazenou orientací. Ta však může obsahovat velké množství úseček, v závislosti na kvalitě segmentace a skeletonizace. Proto je za současného nastavení ponecháno pouze 15% nejdelších úseček, nebo úseček delších než daný práh. Ten je v tuto chvíli nastaven na 34 pixelů. Jedná se další z parametrů, které by bylo možné dále ladit.

Tímto způsobem jsou ponechány pouze dostatečně významné úsečky a daný výsledek je následně normalizován. To znamená, že je spočten obalující box pro danou množinu úseček a ty jsou normalizovány do čtverce o velikosti 200×200 pixelů. Výsledky extrakce stroků pro tagy známé z obr. 5.1 jsou znázorněny na obr. 5.4.

Porovnávání Vstupem porovnání jsou vždy dva tagy, respektive instance klasifikátoru je inicializována dotazem a vstupem porovnávání je pak právě jeden dotazovaný tag. Výsledkem je skóre jejich vzájemné podobnosti. Jak bylo již předestřeno, musí být umožněno porovnávat jak se zdrojovým obrázkem, tak s daty uloženými v souborech. To znamená, že je v případě porovnaní s tagem mimo databázi je ještě nutné provést naskenování stroků, místo pouhého načtení ze souboru.

Porovnání tedy spočívá zjednodušeně ve zjištění, kolik úseček oba tagy sdílejí. Je tedy v praxi nutno porovnat dva neseřazené seznamy, složené z trojic (počáteční bod, koncový bod, orientace). Znamená to tedy projít seznam trojic, kterými je tvořen první tag a zjistit, kolik procent z nich se nachází v tagu druhém. Následně toto provést v obráceném gardu, tedy zjistit kolik těchto trojic z druhého tagu se nachází v prvním tagu. Výsledkem je pak menší číslo z dvojice mezivýsledků.

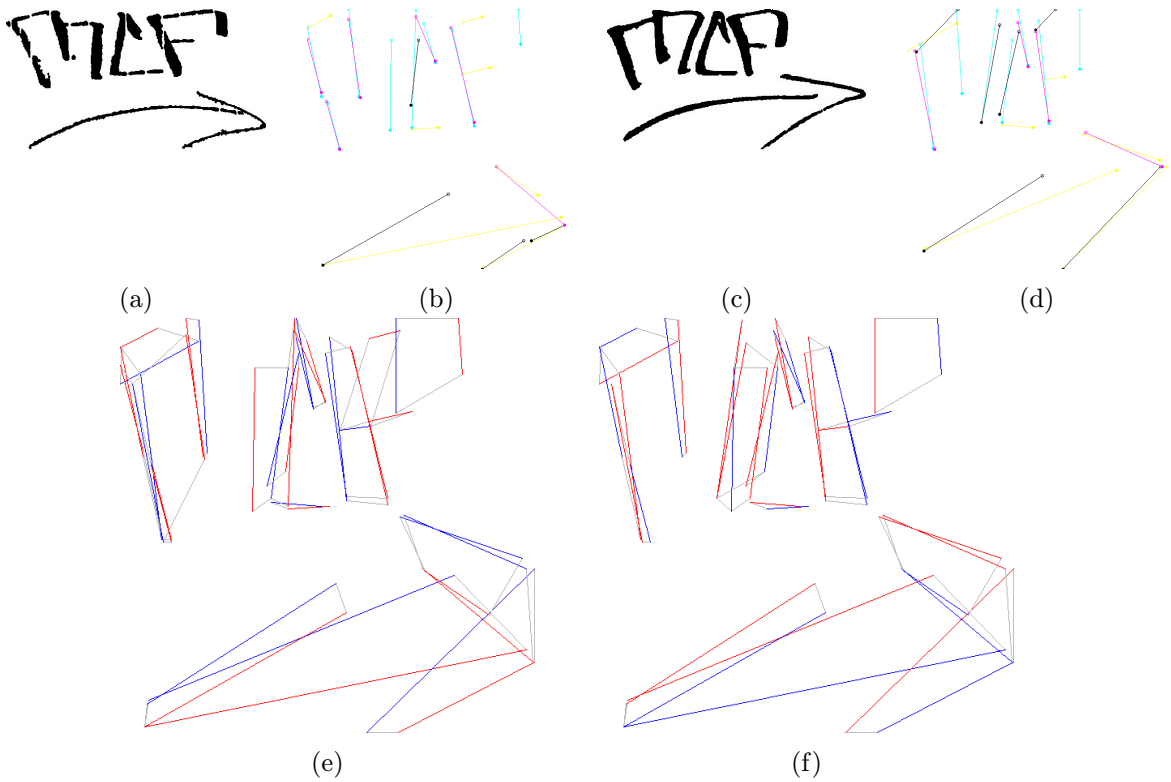
Pro účely porovnávání je však tento způsob reprezentace nevhodný, neboť by takové porovnání mělo kvadratickou časovou složitost v závislosti na počtu porovnávaných strokes. Z tohoto důvodu byl celý problém transformován na vyhledávání v n–dimenzionálním prostoru. Transformace probíhá tak, že:

- **počáteční a koncový bod** jsou dekomponovány na x a y souřadnice, tedy (x_1, y_1, x_2, y_2),
- **orientace** na dvousložkový vektor podle tabulky 5.2, tak aby bylo možné penalizovat přeskok na jinou orientaci.

Tím je tedy získán šestimístný vektor $(x_1, y_1, x_2, y_2, o_1, o_2)$. Před porovnáním jsou oba tagy převedeny do této reprezentace. Tento způsob umožňuje libovolné nastavení pro penalizaci `ORIENTATION_WEIGHT` při přechodu mezi orientacemi. V mezních situacích totiž dochází k zařazení jednoho stroku ke dvěma orientacím. Tímto způsobem je umožněno experimentování s hodnotou penalizace pro změnu orientace. `ORIENTATION_WEIGHT` je tady další z laditelných parametrů.

Tabulka 5.2: Transformace orientace do vektoru

Směr hrany	Přiřazený vektor
horizontální	(`ORIENTATION_WEIGHT`, 0)
vertikální	(`-ORIENTATION_WEIGHT`, 0)
45°	(0,`-ORIENTATION_WEIGHT`)
135°	(0,`ORIENTATION_WEIGHT`)



Obrázek 5.6: a) Dotazovaný tag; b) Strokes dotazovaného tagu; c) nejpodobnější tag v databázi; d) strokes nejpodobnějšího tagu; e) a f) vzájemné výsledky porovnání těchto tagů.

Samotné porovnání pak proběhne tak, že je jeden z tagů, tvořený množinou orientovaných stroků, převeden dle výše uvedeného postupu do 6–dimenzionálního vektoru. Nad

množinou je vystavěn KD strom, který umožňuje efektivně provádět prostorové dotazy. Pro každý orientovaný stroke z prvního tagu je posléze vyhledán nejbližší stroke s využitím KD stromu. Z důvodu optimalizace je při vyhledávání použita jako metrika druhá mocnina eukleidovské vzdálenosti, tzv. *euclidean squared*. Ta však není metrikou (nesplňuje podmínku trojúhelníkové nerovnosti), proto je po získání výsledku odmocněna, čímž je získána eukleidovská vzdálenost definovaná obecným vzorcem 5.1 mezi těmito dvěma stroky. Pokud je nižší, než je stanovený práh (laděný parametr `DIST_THRESH`) je výsledek prohlášen za shodu a algoritmus pokračuje vyhledáním dalšího stroku v pořadí. Na závěr je podle výše zmíněného postupu spočteno závěrečné skóre, které je vráceno jako výsledek porovnání a nabývá rozsahu $\langle 0.0, 1.0 \rangle$.

$$d(\vec{a}, \vec{b}) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \tag{5.1}$$

5.5 Porovnávání projekcí

Vstupem pro tento algoritmus je binární obraz tagu. Ačkoliv je v článku [16] prováděno extrahování projekcí ze skeletonizovaného obrazu, bylo od tohoto způsobu po testování upuštěno a je používán pouze segmentovaný obraz tagu.

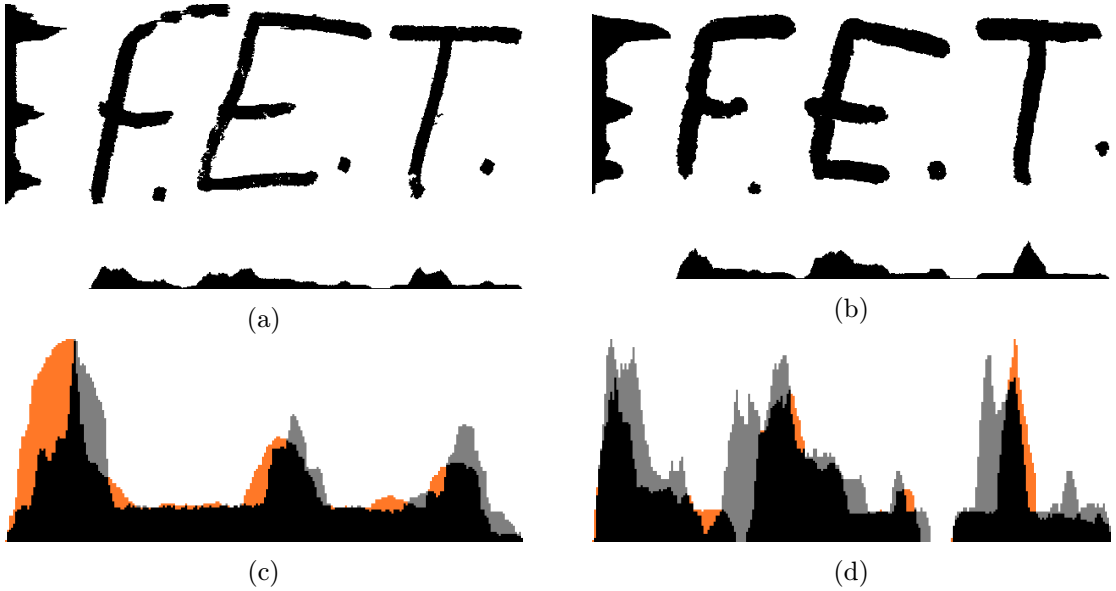
Inicializace Nejprve jsou ořezány okraje obrazu neobsahující popředí. Následně jsou spočteny histogramy projekcí pro obě dimenze, které jsou uloženy do dvou polí. Na obr. 5.6a a 5.6b. V dalším kroku je provedeno normalizování obou histogramů na velikost 256 binů pro každou dimenzi. Taktéž jsou normalizovány hodnoty v každém z histogramů na rozsah $\langle 0.0, 1.0 \rangle$. Tento postup není v původním článku aplikován z důvodu, že je jako vstup používán skeletonizovaný obraz. V tomto případě je však normalizaci nutno provést kvůli rozdílné šířce linií tagů. Díky tomu je tento postup nezávislý na aktuální šířce použitých čar, ale zohledňuje při porovnávání průběh šířky linie, což je žádoucí efekt.

Takto získaná data je již možné uložit do souboru stejným způsobem jako u porovnávání pomocí strokes. Oba dva histogramy, horizontální i vertikální, jsou uloženy do souboru formátu `YAML v. 1.0`.

Porovnávání Porovnání dvou projekcí spočívá ve spočtení kumulovaného rozdílu jednotlivých binů histogramů. Pro zajištění odolnosti oproti drobnému vzájemnému posuvu obou projekcí je provedeno vzájemné posunutí o N polí oběma směry. Jako výsledek je vybrána nejlepší shoda v obou porovnávaných dimenzích, tato čísla jsou sečtena a normalizována na hodnotu v rozsahu $\langle 0.0, 1.0 \rangle$. Porovnání dvou normalizovaných projekcí bez vzájemného posuvu je vyobrazeno na obr. 5.6c, respektive 5.6d, kde je vyobrazeno porovnání normalizovaných horizontálních a vertikálních projekcí pro tagy na obr. 5.6a a 5.6b.

5.6 Metody pro hrubé porovnávání

Tyto metody by měly umožnit předběžné vyhledávání v případě, že se v databázi bude nacházet větší množství dat a podrobné porovnání nebude možné. Hlavním požadavkem je rychlost porovnání. Jako nejvýznamnější zefektivnění vyhledávání je považováno prohledávání databáze podle lokality, v tuto chvíli však není tato funkcionalita na pořadu dne, neboť



(a) (b) (c) (d)

Obrázek 5.7: a) a b) Dva podobné tagy a jejich projekce před normalizací; c) porovnání normalizovaných horizontálních projekcí těchto tagů; d) porovnání normalizovaných vertikálních projekcí těchto tagů.

testovací data nejsou anotována pomocí GPS, navíc toho mapování probíhalo v rámci Brna a tedy by tento způsob zefektivnění neměl valného smyslu.

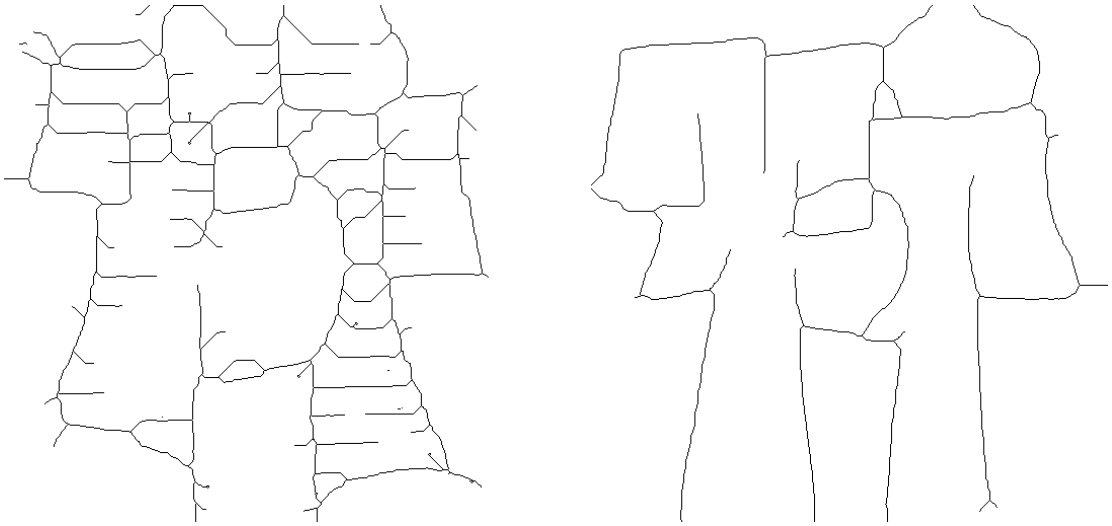
Poměr stran Implementace tohoto porovnání je velmi jednoduchá. Na vstupním obraze je spočten obalující box pro vstupní tag a z něho je následně spočten poměr stran. Stejně jako ostatní klasifikátory v této kategorii nemá výstup normalizovaný na hodnoty v rozsahu $\langle 0.0, 1.0 \rangle$, neboť toto není z principu možné. Pro výstup platí pouze horní omezení intervalu, kde skóre 1.0 mohou získat dva tagy s totožným poměrem stran. Výsledek porovnání však může být i záporný a teoreticky není zdola omezen, proto je nutné s tímto počítat při případném využití spolu s dalšími klasifikátory.

Počet křížení a počet koncových bodů Tyto dva algoritmy byly zavrženy již během implementace, neboť bylo zjištěno, že počty jak koncových bodů, tak křížení, se ve skeletonizovaném obraze velmi odlišují v rámci jednotlivých tříd. Nejzávažnějším problémem se jeví být prakticky nulová odolnost oproti šumu a artefaktům ve skeletonizovaných obrazech.

Příkladem budiž situace na obrázcích 5.7a a 5.7b, kde dojde ke znásobení počtu koncových bodů i křížení. Prioritou algoritmů pro hrubé porovnání by mělo být neodstranění relevantních dat, což by ve zmíněném případě bylo porušeno. Pro úplnost je nutno uvést, že kombinace algoritmu porovnávání pomocí strokes a projekcí si s porovnáním těchto dvou tagů poradí.

Počet maxim na projekčním histogramu Tato metoda je implementována v rámci klasifikátoru porovnávajícího projekce, neboť využívá jeho funkcionalitu a soubor s příznaky. Souřadnice těžiště byly získány z normalizovaného projekčního histogramu. Jsou tedy normalizované na rozsah $\langle 0.0, 200 \rangle$ na obou osách.

Maxima na projekčních histogramech jsou spočtena ve dvou průchodech. Nejprve jsou



(a) (b)

Obrázek 5.8: a) artefakty zasažený exemplář tagu; b) jiný exemplář tagu stejné vizuální třídy.

z pole hodnot odstraněny duplicitní hodnoty a druhým cyklem jsou spočtena maxima. Tato dvojice čísel je spolu se souřadnicemi těžiště uložena do souboru s příznaky pro další použití.

Porovnávání dvou tagů je pak realizováno jako součet absolutních hodnot rozdílů v počtech horizontálních a vertikálních maxim. Výsledek porovnání je pak odečten od konstanty. Tato úprava výsledku byla použita, aby nebylo nutné zasahovat do jádra frameworku.

Souřadnice těžiště Tato metoda byla zařazena do implementace porovnávání pomocí projekcí. Souřadnice je vypočtena jako průměrná souřadnice bodů tvořících popředí při výpočtu obalového tělesa tagu. Porovnání je realizováno výpočtem eukleidovské vzdálenosti mezi dvěma těžišti a obdobnou úpravu jako v případě předchozí metody.

Počet stroků jednotlivých orientací Tento způsob porovnání dvou tagů je implementován v rámci metody porovnávání pomocí strokes, neboť ke svému běhu využívá metody tohoto klasifikátoru. Porovnání je založeno na sumě absolutních hodnot rozdílů jednotlivých složek, tedy počtu vertikálních, horizontálních a úhlopříčných stroků.

5.7 Skripty pro ladění parametrů

Ladění parametrů se skládá ze dvou fází, trénovací a testovací. Dataset je rozdělen tak, že na tréning jsou použity třídy, které mají malý počet členů. V tuto chvíli je hranice nastavena na počtu 5 tagů v dané třídě. Má-li třída více členů, je použita na testování.

Rozdělení do tříd je získáváno ze souboru `classes.data`, který se nachází ve složce `classifi/data/queries`. Tento soubor je vytvářen skriptem `create_classes.py`. Tento skript prochází složky v adresáři `classify/classes/` a vytváří z nich jednotlivé třídy určené pro automatickou kontrolu výsledků. Identifikátor třídy je zvolen dle názvu složky, který by měl být pokud možno přepisem daného tagu. Začlenění tohoto skriptu do procesu vytváření datasetu je popsáno v sekci 5.3.

Tento přístup vytváření datasetu je zvolen z důvodu jednoduchosti, neboť není nutné implementovat žádné grafické rozhraní a zároveň je možné velmi snadno vizuálně kontrolovat a samozřejmě měnit zařazení do jednotlivých tříd, neboť operační systémy běžně umožňují zobrazení náhledů obrázků. Protože se však jedná o přesouvání obrázků mezi složkami, mohou snadno vzniknout duplicitní výskyty v různých třídách, případně jiné porušení integrity datasetu. Z tohoto důvodu byl implementován skript `integrity_check.py`, který se nachází ve stejné složce jako soubor `classes.data`. Výsledkem jsou tři seznamy nalezených chyb v integritě datasetu:

- tagy vyskytující se jen provozním datasetu,
- tagy vyskytující se pouze ve složce `classes`,
- tagy zařazené do více tříd.

Možnosti ladění Hlavní funkcionalitu pro ladění parametrů poskytuje skript `tuner.py`, který je umístěn ve složce `classify/bin`. Jedná se o složku s binárními soubory klasifikátorů. Tento skript umožňuje spouštět vyhledávání s různými parametry a zjišťovat tak Samotné ladění parametrů lze rozdělit na dvě části.

První částí je ladění **parametrů jednotlivých klasifikátorů**. Při něm jsou generovány hodnoty v daných definovaných rozsazích. Je možné specifikovat, zda budou generována čísla datového typu `integer`, nebo `float`. Tyto parametry jsou ukládány do konfiguračních souborů jednotlivých klasifikátorů, odkud jsou načítány při inicializaci klasifikátoru. Původně bylo uvažováno o předávání jednotlivých parametrů příkazovou řádkou, nicméně tato volba se ukázala být nekoncepční, neboť by bylo nutné zasahovat do rozhraní jednotlivých klasifikátorů.

Druhým krokem je **ladění vah jednotlivých klasifikátorů**. V této fázi je v jedné iteraci každému klasifikátoru nastavena váha v rozsahu $\langle 0.0, 1.0 \rangle$. Suma těchto vah se musí jednat jedné, je tedy nutné po náhodném generování provést normalizaci.

Proces ladění probíhá po cyklech, jejichž počet lze definovat ve funkci `main` skriptu `tuner.py`. Během každé iterace dojde k předání úlohy frameworku. Jako dotaz je položen vždy první tag v dané třídě. Framework provede vyhodnocení a vrátí na standardní výstup celkový výsledek. Ten je skriptem vyhodnocen oproti souboru `classes.data`.

Jako základní metrika je zvolena *Mean Average Precision*. Ta je vypočítána jako průměr hodnot *Average Precision* jednotlivých dotazů. *Average Precision* lze nejlépe vyjádřit vzorcem **??**, kde R_{good} je počet správných výsledků a R_{total} je celkový počet vrácených položek do vrácení všech správných výsledků.

$$AveP = \frac{R_{total}}{R_{good}} \tag{5.2}$$

Tato metrika může vykazovat horší hodnocení v případech, kdy testovaná třída obsahuje tag, který není dle použitých klasifikátorů podobný dotazu(jedná se o tzv. *outlier*). Z tohoto důvodu je vhodné kontrolovat výsledky pomocí vykreslení křivky *precision–recall*, která vyjadřuje průběh hodnoty *Average Precision* v závislosti na počtu vrácených výsledků. Výsledky jednotlivých včetně hodnot aktuálně laděných parametrů jsou logovány do souboru *output.log* ve složce `data/results`.

Obvyklým postupem je nejprve vyladit parametry jednotlivých klasifikátorů jako celků a poté ladit jejich vzájemné vyvážení. V obou případech je vhodné zpočátku nechat parametry měnit hodnoty volně v rámci rozsahu hodnot, který vyplývá z implementace. Následně

prozkoumat vliv jednotlivých složek na výslednou hodnotu, upravit nastavení rozsahů, případně některé hodnoty zmrazit. Nalezenou kombinaci parametrů je nutné ověřit experimentem na testovací části datasetu.

5.8 Uživatelské dotazy na databázi

Za účelem uživatelského testování algoritmů byl vytvořen skript, který umožňuje spouštět vyhledávací proces pomocí *drag&drop* přímo z prostředí operačního systému. Stačí nakopírovat segmentovaný snímek tagu do složky `classify/data/queries` a přetáhnout ho na skript `SEARCH.bat`. Tento dávkový soubor pouze předá přetažený snímek jako parametr skriptu `search.py` v téže složce.

Ten zajistí předzpracování v souladu s procedurou na přípravu dat 5.3. Obrázek s tagem je tedy upraven na velikost 512×512 pixelů a je mu vytvořen skeleton. Následně je spuštěn porovnávací proces pomocí frameworku. Výsledky jsou zapsány do souboru `result.html` a ten je následně spuštěn pomocí defaultního prohlížeče v systému. Zobrazeno je prvních dvacet výsledků ve výsledkové listiny, doplněných o výsledné skóre.

Kapitola 6

Experimenty a vyhodnocení

V této kapitole jsou popsány provedené experimenty a jejich výsledky, a to jak z hlediska jednotlivých klasifikátorů, tak z hlediska výsledné funkcionality aplikace. Dále je zde rozebrána praktická využitelnost výstupů této práce a alternativní využití implementovaných algoritmů. Taktéž je popsán výsledný dataset.

6.1 Dataset

Experimenty byly prováděny na postupně budovaném datasetu. Během jeho rozšiřování bylo zjištěno několik zajímavých poznatků z reálií graffiti scény, které jsou důležité z hlediska sběru dat. Při shromažďování dat byla potvrzena domněnka, že se stejné tagy vyskytují ve shlucích. Mapování komplexního území bez dřívějšího plánování vede k získání velkého množství dále nevyužitelných dat, tedy vizuálních tříd, které jsou unikátní nebo se vyskytují pouze v malých počtech a k vývoji a testování klasifikátoru nejsou moc použitelné. Případně se velmi často jedná o rozšířené, ale již velmi staré tagy (podle zprávy[1] společnosti GRIP Systems je až 70% graffiti ve městech starých 5-10 let), které budou v okolí již přemazány, případně se budou vyskytovat ve více verzích, což způsobí jejich nevhodnost pro účely vývoje systému.

Další komplikací je průběžný vývoj, kterým tagy od jednoho autora časem prochází. Je tedy možné získat poměrně velké množství tagů např. sbíráním dat na sídlištích, ale je možné a pravděpodobné, že se bude jednat o vzorky z různých časových období. Jak již bylo zmíněno dříve v textu, sprejeři se zaměřují na veřejně exponovaná místa. Proto je možné k počátečnímu průzkumu využít prostředky hromadné dopravy a následný sběr uskutečnit již v předem prozkoumané oblasti. Tím je možné zásadně zvýšit efektivitu pořizování dat.

Celkem je v tuto chvíli v databázi nasbíráno **417** segmentovaných graffiti tagů. Pro účely testování jsou zařazeny do **32** tříd. **11** z těchto tříd obsahuje má **6** a více členů. Tyto třídy jsou používány především jako testovací po odladění klasifikátoru. Zbytek tříd obsahuje méně než **6** členů a tyto třídy jsou používány pro tréning, respektive ladění jednotlivých parametrů. Nejpočetnější je třída označená pracovně přepisem *FI*, která obsahuje **25** členů. Celkem **230** tagů není zařazeno do žádné třídy a slouží tedy spíše pro otestování robustnosti algoritmu.

[1] Zdroj: http://www.gripsystems.org/pages/fact.pdf

6.2 Vlastnosti porovnávání pomocí strokes

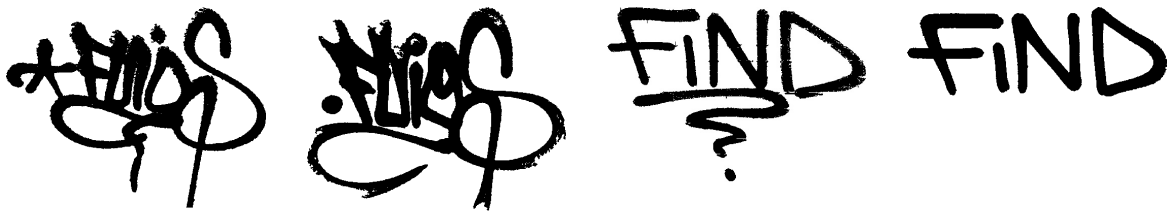
Základní vlastnosti tohoto algoritmu vyplývají již z jeho implementace. Porovnávání pomocí strokes je robustní vůči:

- **změně měřítka,**
- **změně poměru stran,**
- **různému průběhu a tloušťce linií.**

Dále z implementace plyne, že je náchylný vůči

- **změně perspektivy**,
- **otočení.**

Další praktické omezení využitelnosti algoritmu plyne z faktu, že jsou vždy porovnávány dva tagy jako celky. Nelze např. vyhledávat pouze podle jejich charakteristických částí. Tento algoritmus je částečně odolný oproti různým *ozvláštněním* daného tagu, zejména pokud nepřesahují obalující těleso jádra tagu. Za tímto účelem jsou writery používány různé vlnovky, podtržení, interpunkční znaménka případně různé další prvky. V případě, že tyto úpravy zasahují výrazně mimo hranice obalujícího obdélníku, bude výsledek silně negativně ovlivněn. Tyto úpravy jsou dobře pozorovatelné na obrázcích 6.1a až 6.1d.



(a) (b) (c) (d)

Obrázek 6.1: a) až d) Použití různých kaligrafických prvků.

K podobným problémům může vést i následkem nedokonalé segmentace, kdy dojde ke sloučení s jiným tagem nebo naopak odstranění významné části tagu. Z tohoto plyne nemožnost vyhledávat podle charakteristických znaků písma, jako je styl psaní konkrétních písmen nebo jiný významný prvek vyskytující se v daném tagu. Z hlediska použití v praxi je toto bohužel poměrně zásadní omezení funkcionality.

Výsledky Parametry pro výsledné použití byly nastaveny následujícím způsobem:

`DIST_THRESH`**: 53** Tento parametr vyjadřuje maximální eukleidovskou vzdálenost mezi dvěma stroky v 6–dimenzionálním prostoru.

`ORIENTATION_WEIGHT`**: 300** Toto je penalizace pro přeskok na stroke s jinou orientací. Tato hodnota ve spojení s hodnotou parametru `DIST_THRESH` vyjadřuje, že při porovnání nedojde k označení dvou (byť blízkých) stroků s různou orientací jako shodných.

Celkový výkon (tedy *Mean Average Precision*) tohoto klasifikátoru za dané konfigurace vyjádřený číslem činí **0,31** na trénovací sadě a **0,34** na testovací sadě.

6.3 Porovnávání projekcí

Podobně jako porovnávání pomocí strokes je tento algoritmus odolný oproti:

- **změně měřítka,**
- **změnám poměrů stran**,
- **tloušťce linií.**

Naopak jako odlišné vyhodnotí tagy, které:

- **mají různé průběhy šířky linií,**
- **jsou navzájem natočeny,**
- **jsou vyfoceny z jiné perspektivy.**

Taktéž se jedná o algoritmus, který pracuje s tagem jako celkem se všemi důsledky z toho vyplývajícími. Oproti algoritmu porovnávání pomocí strokes nabízí především nezávislosti na skeletonizaci tagu, čímž je i podstatně robustnější vůči případnému šumu a zároveň umožňuje odlišit tagy podle změny průběhu linií.

Tato metoda obsahuje jediný laditelný parametr, šířku porovnávacího okna (výše v textu označena jako **N**). Nejlepší výsledky poskytuje tento způsob porovnávání za nastavení $N = 18$. To tedy znamená, že je jsou projekce vzájemně posunuty o 18 pixelů oběma směry, což poskytuje vyšší robustnost např. oproti prodloužení koncových tahů.

Celkově tedy dosahuje tato metoda hodnot Mean Average Precision **0,19** pro trénovací, respektive **0,14** pro testovací dataset.

6.4 Metody pro hrubé porovnávání

Pro vyhodnocení těchto výsledků byl použit stejný princip, jako u dvou hlavních klasifikátorů, tedy výpočet hodnoty Mean Average Precision. Z výsledků testování vyplynulo, že jsou tyto způsoby porovnávání jsou celkem nepřesné a snadno mohou zavrhnout některý ze správných výsledků. Proto by mohly být využity k odstranění tagů na první pohled nevhodných, tzn. tagy mající diametrálně odlišný poměr stran, nebo počty stroků. Výsledky byly zaznamenány do tabulky 6.1. Uváděny jsou zvláště výsledky pro trénovací dataset a testovací dataset.

Tabulka 6.1: Výsledky jednotlivých metod pro hrubé porovnání.

Metoda	MAveP testovací dataset	MAveP trénovací dataset
Poměry stran	0,06	0,17
Počty stroků	0,12	0,08
Pozice těžiště	0,06	0,12
Náhodné řazení	0,03	0,01

Pro porovnání je v tabulce uveden i výsledek v případě náhodného řazení výsledků. Za povšimnutí stojí např. výsledek porovnání projekcí na trénovacím datasetu. Tento výsledek je možné vysvětlit tím, že trénovací třídy jsou tvořeny menším počtem členů, které nemají

tak velkou variabilitu v rámci tříd. Tuto hypotézu podporuje již o poznání horší výsledek na testovacím datasetu. Tento efekt zřejmě nastal i v případě porovnávání vzdáleností těžišť.

Hlavní výhodou těchto metod je fakt, že by mohly být implementovány přímo v prostředí relační databáze, čímž by šlo efektivně vyřadit část výsledků a šetřením procesorového času serveru, který by hostil primární porovnávací algoritmy.

6.5 Celkové výsledky

Dva výše jmenované klasifikátory, porovnávání pomocí strokes a porovnávání projekcí, byly propojeny za účelem dosažení lepšího výsledku. Původně bylo testováno i přičlenění porovnávání podle poměrů stran, avšak k vylepšení celkového výsledku nedošlo. V současné konfiguraci tak má klasifikátor porovnávající pomocí strokes váhu **0,45** a projekce **0,55**. Takto propojené klasifikátory dosahují hodnoty Mean Average Precision **0,37** na trénovací sadě, respektive **0,40** na testovací sadě.

Vývoj hodnoty přesnosti v závislosti na počtu vrácených prvků znázorňuje křivka *Precision–Recall*. Na ose X je vynesen počet aktuálně vrácených prvků. Záznam na této ose končí v okamžiku, kdy jsou vráceny všechny správné výsledky. Na osy Y je vynesena aktuální hodnota přesnosti.

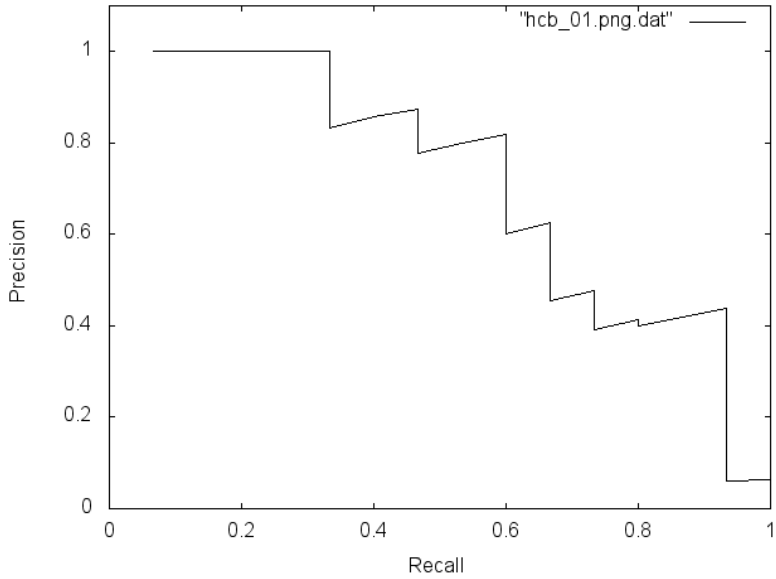
Na grafu 6.2 je znázorněn průběh křivky *Precision–Recall* pro třídu , která má průměrnou úroveň variability jednotlivých prvků. Hodnota *Average Precision* – 0,07 – je přitom na první pohled nízká. Při bližším zkoumání grafu je možné dovodit, že se jedná o jediný outlier. V tomto případě jsou na obrázcích 6.3a až 6.3c znázorněny tři běžné tagy z této třídy a na obr. 6.3d je zobrazen poslední nalezený člen, v tomto případě tedy outlier, který se ve výsledku nacházel až na 233. místě.

Graf 6.4a vyjadřuje průběh křivky *Precision–Recall* pro třídu s minimální variabilitou prvků, což je výjimečný případ. Na obrázcích 6.5a až 6.5h jsou pro názornost vyobrazeny 4 prvky této třídy. Jak je z obrázků patrné, jednotlivé tagy se od sebe příliš neliší.

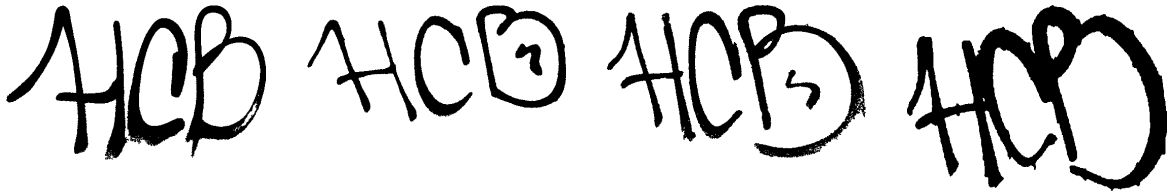
Opačný extrém je možné vidět na grafu 6.4b. Třída jejíž prvky jsou vyobrazeny na obr. 6.5d až 6.5g je zasažena poměrně velkou variabilitou prvků, čímž negativně ovlivňuje výsledek vyhledávání členů této třídy. Všechny tyto tři třídy jsou pro úplnost celé vyobrazeny v přílohách.

Praktická použitelnost Závěrem je možné říci, že výsledky dosažené pomocí popsaných algoritmů by mohly významně usnadnit práci při organizování velké databáze graffiti tagů. Nicméně z důvodu poměrně vysokých požadavků na předzpracování vstupních obrázků, z toho plynoucí závislosti na lidském faktoru a nemožnosti vyhledávat podle části tagu se jeví jako do budoucna lepší přístup zkoumaný Vojtěchem Grünseisenem v jeho diplomové [8]. Ten se zakládá na metodách užívaných spíše k vyhledávání podobných obrázků a poskytuje minimálně srovnatelné výsledky, ovšem bez potřeby složitého předzpracování. Taktéž z principu umožňuje i porovnávat podle charakteristických částí tagu. Navíc se ve vyhledávání neomezuje na tagy, ale umožňuje vyhledávat a porovnávat i jiné formy graffiti.

Pro případné praktické nasazení by bylo nutné vhodným způsobem vyřešit především vstupní bod do systému tak, aby byla zaručena dodávka kvalitně předzpracovaných dat. Také by bylo nutné umístit porovnávací software na server, kde by mohl obsluhovat požadavky klientů. Z důvodu zmíněných výše však další pokračování ve vývoji těchto konkrétních metod není v tuto chvíli plánováno.

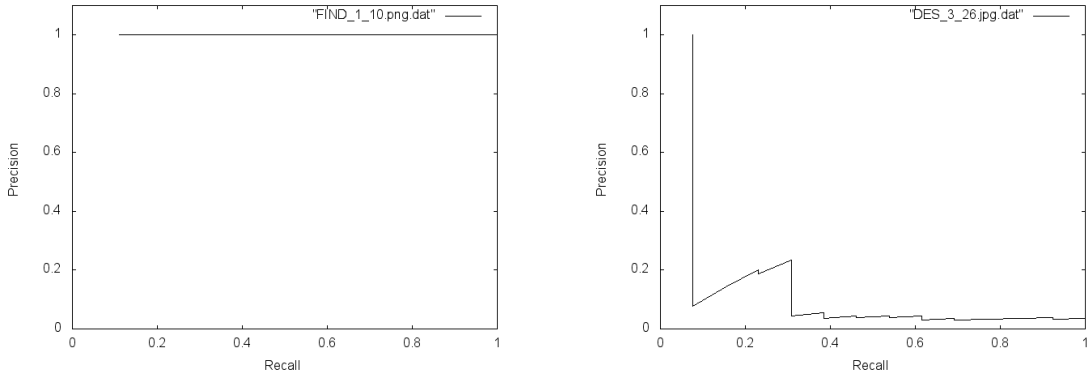


Obrázek 6.2: Průběh křivky precision–recall průměrný výsledek vyhledávání.



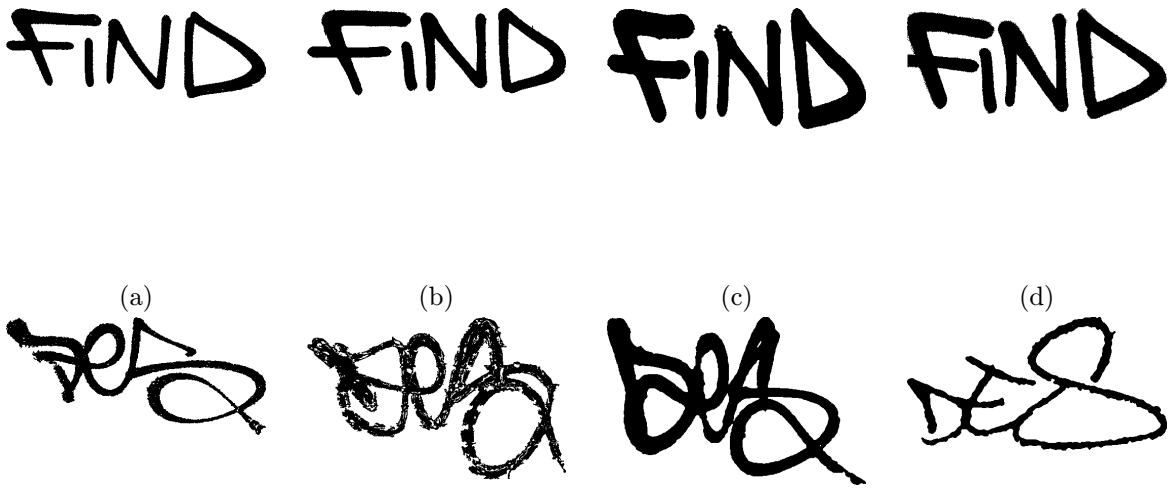
(a) (b) (c) (d)

Obrázek 6.3: a) Vyhledávaný tag; b) nejlepší shoda; c) další prvek třídy; d) outlier.



(a) (b)

Obrázek 6.4:) Průběh křivky precision–recall pro výjimečně dobrý (a)) a špatný (b)) výsledek.



(a) (b) (c) (d)

(e) (f) (g) (h)

Obrázek 6.5: a), e) Vyhledávaný tag; b), f) nejlepší shoda; c), g), h) další prvky tříd; d) outlier.

Kapitola 7

Závěr

V této práci se mi podařilo adaptovat přístupy používané k rozpoznávání podpisů na problematiku vyhledávání podobných graffiti tagů. Ačkoliv jsou tyto metody v případě vyhledávání tagů znatelně méně úspěšné, než při rozpoznávání podpisů, jejich nasazení by již citelně zefektivnilo správu databáze graffiti.

Vypracoval jsem metodiku pro hromadný sběr dat a v souladu s ní nasbíral, předzpracoval a anotoval přes 400 tagů. Tato data je možné využít k dalšímu vývoji a testování.

Dále jsem zdokumentoval technické rozdíly mezi graffiti tagy a získal přehled o základních principech fungování graffiti subkultury. Tyto informace dále využijeme v rámci další práce na projektu TagBust. Ten se v tuto chvíli nachází ve stavu testovacího nasazení za účelem sběru dat městskou policií v Břeclavi. Další kroky našeho týmu tedy povedou k rozšíření aktuálních funkcionalit systému o algoritmy, které jsme zpracovali v našich diplomových pracích. Také máme v plánu brzy uveřejnit aplikaci pro sběr dat veřejností. Tuto aplikaci zpracoval a ve svojí práci [12] popisuje Matěj Kubiš.

Z hlediska budoucího nasazení se v tuto chvíli jeví jako více perspektivní způsob vyhledávání, který v rámci svojí diplomové práce [8] zkoumal můj kolega, Vojtěch Grünseisen. Jím zkoumaná metoda umožňuje porovnávat nejen tagy, ale i ostatní formy graffiti. Toto navíc zvládne bez potřeby segmentace obrazu, která se ukázala být velmi náročnou úlohou. Tato metoda si zároveň poradí s detekováním duplicitních záznamů v databázi, což je pro efektivní provoz takové databáze nezbytné.

Literatura

[1] Simple Interactive Object Extraction. [cit. 7.1.2013].
URL http://www.siox.org/

[2] Castleman, C.: *Getting Up: Subway Graffiti in New York*. MIT Press, 1982, ISBN 9780262030892.

[3] Chatbri, H.; Kameyama, K.: Towards making thinning algorithms robust against noise in sketch images. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, IEEE, 2012, ISBN 978-1-4673-2216-4, s. 3030–3033.

[4] Comaniciu, D.; Meer, P.: Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 24, č. 5, 2002: s. 603–619, ISSN 0162-8828.

[5] Das, R.: An introduction to signature recognition as a biometric technology. *Keesing Journal of Documents & Identity*, 2007: s. 603–619, ISSN 1571-0564.

[6] Drahanský, M.; Orság, F.; Doležel, M.; aj.: *Biometrie*. Computer Press, první vydání, 2011, ISBN 978-80-254-8979-6, 294 s.

[7] Eakins, J.; Graham, M.; Franklin, T.: Content-based image retrieval. In *Library and Information Briefings*, Citeseer, 1999.

[8] Grünseisen, V.: *Vyhledávání graffiti tagů podle podobnosti*. Diplomová práce, FIT VUT v Brně, 2013.

[9] Hartigan, J. A.; Wong, M. A.: Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, ročník 28, č. 1, 1979: s. 100–108, ISSN 0035-9254.

[10] Hlaváč, V.: Matematická morfologie. [cit. 18.5.2013].
URL http://cmp.felk.cvut.cz/~hlavac/Public/TeachingLectures/BinMatMorfolCesky.pdf

[11] Karki, M.; Indira, K.; Selvi, S.: Off-Line Signature Recognition and Verification Using Neural Network. In *Conference on Computational Intelligence and Multimedia Applications.*, ročník 1, dec. 2007, s. 307 –312, doi:10.1109/ICCIMA.2007.296.

[12] Kubiš, M.: *Mobilní a webová aplikace pro sběr a správu graffiti*. Diplomová práce, FIT VUT v Brně, 2013.

[13] Ozden, M.; Polat, E.: Image segmentation using color and texture features. In *Proceeding of the 13th European Signal Processing*, 2005, ISBN 9781604238211, s. 2226–2229.

[14] Özgündüz, E.; Şentürk, T.; Karsligil, M.: Off-line signature verification and recognition by support vector machine. In *European Signal Processing Conference.*, 2005, ISBN 9781604238211, s. 2370–2073.

[15] Perez-Hernandez, A.; Sanchez, A.; Velez, J. F.: Simplified Stroke-based Approach for Off-line Signature Recognition. In *2nd COST Workshop on Biometrics on the Internet: Fundamentals, Advances and Applications*, Ed. Univ. Vigo, s. 89–94.

[16] Porwik, P.: The Compact Three Stages Method of the Signature Recognition. In *CISIM*, 2007, ISBN 0769528945, s. 282–287.

[17] Tcheslavski, G. V.: Morphological Image Processing: Basic Algorithms. 2009, [cit. 16.5.2013].
URL http://www.ee.lamar.edu/gleb/dip/10-2%20-%20Morphological%20Image%20Processing.pdf

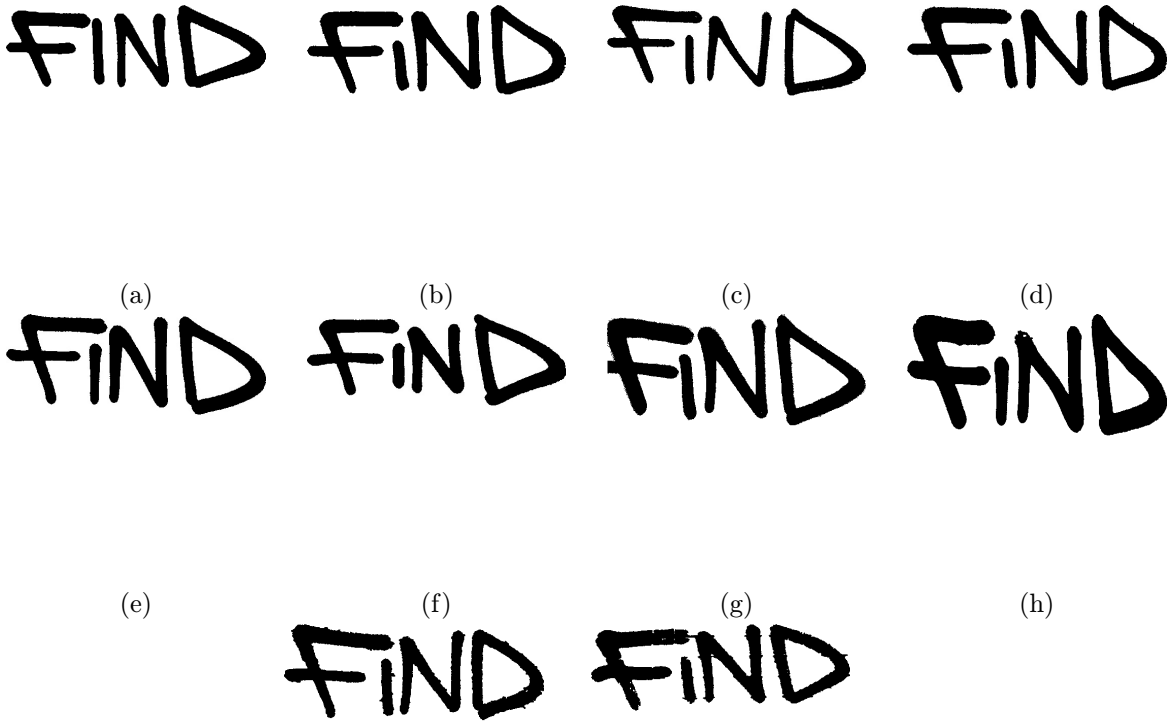
[18] Unser, M.: Texture classification and segmentation using wavelet frames. *IEEE Transactions on Image Processing*, ročník 4, č. 11, 1995: s. 1549–1560, ISSN 1057-7149.

[19] Vondráková, D.: Street art a graffiti (vandalství vs. umění). 2006.

[20] Zacpalová, P.: Graffiti a street art v Brně. 2008.

Příloha A

Variabilita prvků ve třídách



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j)

Obrázek A.1: Třída s výjimečně nízkou variabilitou prvků (*FIND*).

(a) (b) (c) (d)

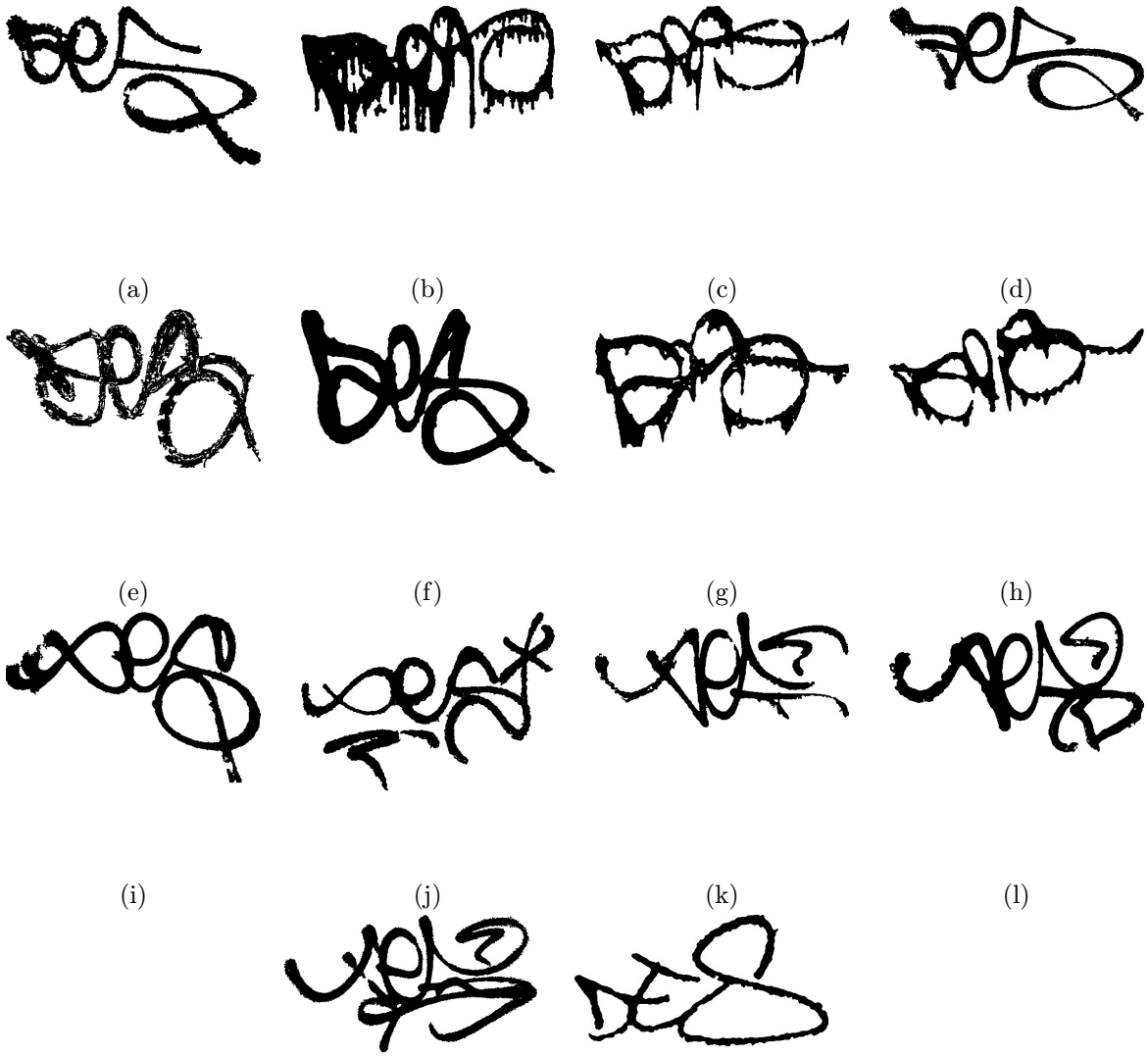
(e) (f) (g) (h)

(i) (j) (k) (l)

(m) (n) (o) (p)

(q)

Obrázek A.2: Třída *HCB*.



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

(m) (n)

Obrázek A.3: Třída s velkou variabilitou členů (*DES*).

Příloha B

Ukázka výsledku vyhledávání

Obrázek B.1: Prvních 18 výsledků vyhledávání v databázi o více než 400 záznamech.